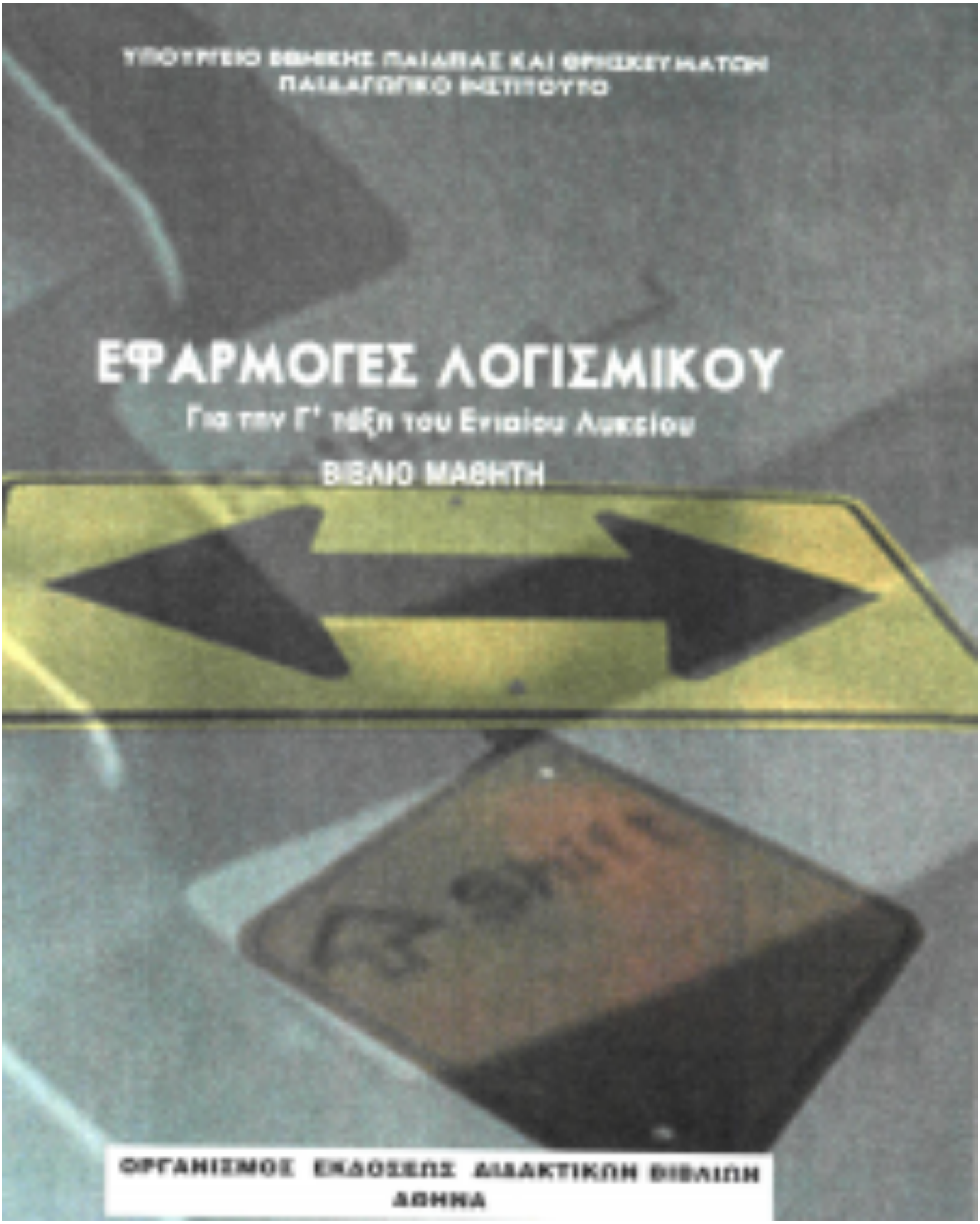
ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ

ΕΦΑΡΜΟΓΕΣ ΛΟΓΙΣΜΙΚΟΥ

Για την Γ' τάξη του Ενιαίου Λυκείου

ΒΙΒΛΙΟ ΜΑΘΗΤΗ



ΟΡΓΑΝΙΣΜΟΣ ΕΚΔΟΣΕΩΣ ΔΙΔΑΚΤΙΚΩΝ ΒΙΒΛΙΩΝ
ΑΘΗΝΑ

ΕΝΟΤΗΤΑ Α'

ΕΝΗΜΕΡΩΝΟΜΑΙ

ΚΕΦΑΛΑΙΟ 1

ΕΠΙΣΚΟΠΗΣΗ ΕΦΑΡΜΟΓΩΝ ΛΟΓΙΣΜΙΚΟΥ

ΠΕΡΙΕΧΟΜΕΝΑ

1.1. Η εξέλιξη των εφαρμογών λογισμικού

1.2. Χαρακτηριστικά και βασικές λειτουργίες των εφαρμογών διαχείρησης δεδομένων

ΕΙΣΑΓΩΓΗ

Όλοι έχουμε χρησιμοποιήσει εφαρμογές λογισμικού με τον ένα ή τον άλλο τρόπο, στο σπίτι, στο σχολείο ή στη διασκέδαση. Οι συσκευές αναπαραγωγής μουσικής από οπτικούς δίσκους, τα ηλεκτρονικά ρολόγια χειρός, οι περισσότερες οικιακές συσκευές, οι κονσόλες ηλεκτρονικών παιχνιδιών (στο σπίτι ή οι μεγάλες με κερματοδέκτη στα καταστήματα) και φυσικά οι ηλεκτρονικοί υπολογιστές, είναι μερικά από τα παραδείγματα συσκευών, που βασίζουν τη λειτουργία τους στο λογισμικό.

- Τι είναι όμως λογισμικό;
- Πόσα είδη λογισμικού υπάρχουν;
- Τι είναι η βάση δεδομένων και πώς λειτουργεί το λογισμικό διαχείρισης δεδομένων;

Σε αυτό το κεφάλαιο θα κάνουμε μια περιήγηση τις εφαρμογές λογισμικού που χρησιμοποιούνται σήμερα, θα δούμε σε ποιες κατηγορίες χωρίζονται και ποια είναι τα χαρακτηριστικά τους, δίνοντας έμφαση στις εφαρμογές διαχείρισης δεδομένων. Ακόμα θα κάνουμε μια εισαγωγή στα μοντέλα δεδομένων, γνώση που θα μας φανεί χρήσιμη στην παρακολούθηση των επόμενων κεφαλαίων.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να διακρίνει τους τομείς χρήσης του λογισμικού στην καθημερινή ζωή.
2. Να κατατάσσει το λογισμικό σε κατηγορίες και να διακρίνει τις εφαρμογές διαχείρισης βάσεων δεδομένων.
3. Να περιγράφει τα γενικά χαρακτηριστικά και τις βασικές λειτουργίες του λογισμικού διαχείρισης βάσεων δεδομένων.
4. Να μπορεί να περιγράψει τι είναι σχεσιακό μοντέλο δεδομένων.

1.1. Η ΕΞΕΛΙΞΗ ΤΩΝ ΕΦΑΡΜΟΓΩΝ ΛΟΓΙΣΜΙΚΟΥ

1.1.1. Τι είναι λογισμικό

Όπως έχουμε μάθει έως τώρα είτε από το σχολείο, είτε πιθανώς από την πείρα μας στη χρήση υπολογιστών, το λογισμικό είναι τα προγράμματα που κατευθύνουν την λειτουργία του υπολογιστή και την επεξεργασία των δεδομένων. Τι εννοούμε όμως λέγοντας "πρόγραμμα";

Ο μεγάλος Ελβετός επιστήμονας Niklaus Wirth, (δημιουργός ανάμεσα στα άλλα και της γλώσσας προγραμματισμού Pascal), το 1975 είχε δώσει τον περίφημο τίτλο σε ένα βιβλίο του:

Προγράμματα = Αλγόριθμοι + Δομές Δεδομένων

Με αυτό ήθελε να πει ότι ένα πρόγραμμα που εκτελείται σε έναν ηλεκτρονικό υπολογιστή είναι ο συνδυασμός μιας προσδιορισμένης με σαφήνεια ακολουθίας εντολών προς τον υπολογιστή (δηλ. ενός **αλγόριθμου**) για την επίτευξη ενός σκοπού και των *δομών δεδομένων* από τις οποίες αντλούν τα δεδομένα που επεξεργάζονται οι αλγόριθμοι και στις οποίες αποθηκεύουν τα αποτελέσματα της επεξεργασίας. Τα δεδομένα αυτά μπορεί να αναπαριστούνται ως μεμονωμένοι αριθμοί ή χαρακτήρες (π.χ., μια αριθμητική μεταβλητή η οποία περιέχει το τραπεζικό επιτόκιο), ή ως συνθέσεις αυτών των απλών τύπων σε πιο σύνθετες δομές, όπως π.χ., ένα πίνακα μίας στήλης και τριών γραμμών από πραγματικούς αριθμούς, στον οποίο αποθηκεύουμε τις καρτεσιανές συντεταγμένες ενός σημείου στο χώρο.

Πέρα όμως από το πρόγραμμα αυτό καθ' αυτό, το λογισμικό είναι και όλο εκείνο το σύνολο των εγγράφων (έντυπων ή ηλεκτρονικών), που περιγράφουν τη χρήση και τις λειτουργίες των προγραμμάτων. Άρα συνοψίζοντας:

Ορισμός

Το **λογισμικό** είναι α) **προγράμματα** που κατευθύνουν την λειτουργία του υπολογιστή και επιτελούν τη διαχείριση και επεξεργασία των δεδομένων, β) τα **έγγραφα** (τυπωμένα ή σε ηλεκτρονική μορφή), που περιγράφουν τη χρήση και τις λειτουργίες των προγραμμάτων.

1.1.2. Κατηγορίες λογισμικού

Είναι δύσκολο να εντάξουμε το σύγχρονο λογισμικό σε κατηγορίες με σαφή όρια μεταξύ τους. Ένα σχετικά προφανές κριτήριο είναι το *περιεχόμενο* του λογισμικού. Ως περιεχόμενο, ορίζουμε τη σημασία (για τον άνθρωπο-χρήστη) και τη μορφή της πληροφορίας που διαχειρίζεται και επεξεργάζεται το λογισμικό.

Για παράδειγμα, στο περιβάλλον μιας εταιρίας ή ενός οργανισμού, θα συναντήσουμε μια πληθώρα εφαρμογών λογισμικού, εγκαταστημένων συνήθως σε υπολογιστές συνδεδεμένους μεταξύ τους σε τοπικό δίκτυο (ή δίκτυα). Άλλες από αυτές ασχολούνται με τη διαχείριση πληροφοριών απαραίτητων για τη διοίκηση της εταιρίας (π.χ., μισθοδοσία, πωλήσεις κ.λ.π.), άλλες

είναι εφαρμογές γραφείου (π.χ., επεξεργασία κειμένου, λογιστικά φύλλα, διαχείριση συναντήσεων κ.λ.π.), άλλες είναι εφαρμογές επικοινωνιών (π.χ., αλληλογραφία, διαμοίραση και κοινή χρήση εκτυπωτών κ.λ.π.), ενώ μπορεί να υπάρχουν και ειδικά εργαλεία ανάπτυξης εφαρμογών (π.χ., γλώσσες προγραμματισμού, συστήματα διαχείρισης βάσεων δεδομένων κ.λ.π.) που χρησιμοποιούνται από τους τεχνικούς και τους επιστήμονες πληροφορικής της εταιρίας για τη συντήρηση και επέκταση των ήδη εγκατεστημένων εφαρμογών λογισμικού (ιδιαίτερα εκείνων που είναι σημαντικές για τη διοίκηση της εταιρίας).

Ένας δυνατός (αλλά όχι αποκλειστικός) τρόπος κατάταξης του λογισμικού σύμφωνα με το περιεχόμενο της πληροφορίας που διαχειρίζεται, είναι ο ακόλουθος:

Ποια προγράμματα από αυτά που έχετε χρησιμοποιήσει έως τώρα εντάσσονται στο λογισμικό συστήματος;

Λογισμικό συστήματος. Αποτελείται από ένα σύνολο προγραμμάτων που έχουν γραφτεί για να υποστηρίζουν άλλα προγράμματα. Τυπικό παράδειγμα είναι τα λειτουργικά συστήματα υπολογιστών. Αυτά ελέγχουν το υλικό *(hardware)* του υπολογιστή, υποστηρίζουν το χειρισμό του υπολογιστή από έναν ή περισσότερους χρήστες, κατανέμουν το χρόνο του επεξεργαστή μεταξύ των προγραμμάτων που είναι ενεργά κ.λ.π. Η διάδοση των τοπικών δικτύων και των δικτύων ευρείας ζώνης (ιδιαίτερα την τελευταία δεκαετία), έχει εμπλουτίσει το λογισμικό συστήματος με το απαραίτητο λογισμικό για την παροχή υπηρεσιών δικτύωσης (π.χ., μεταφορά αρχείων, κοινή χρήση αρχείων και εκτυπωτών κ.λ.π., ασφάλεια δικτύων απέναντι σε ανεπιθύμητους χρήστες κ.λ.π.). Στο λογισμικό συστήματος εντάσσονται και τα εργαλεία ανάπτυξης εφαρμογών όπως οι μεταγλωττιστές (C, Pascal, Cobol, Fortran κ.λ.π.), διάφορα βοηθητικά προγράμματα αποκερματισμού (*defragmentation*) δίσκων, συμπίεσης δεδομένων κ.λ.π. Ακόμη τα τελευταία χρόνια με την εξάπλωση των *Συστημάτων Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ)* που πραγματοποιούν μια κεντρική διαχείριση των δεδομένων για λογαριασμό των εφαρμογών (θα μιλήσουμε για αυτά παρακάτω και στο κεφ. 6), και τον υψηλό βαθμό ολοκλήρωσής τους με τα σύγχρονα εργαλεία ανάπτυξης εφαρμογών, μπορούμε να θεωρήσουμε και τα ΣΔΒΔ μέρος του λογισμικού συστήματος. Άλλωστε πληθαίνουν τα παραδείγματα εταιριών κατασκευής λειτουργικών συστημάτων που είτε προσφέρουν δικά τους ΣΔΒΔ ως τμήμα του λογισμικού συστήματος (π.χ., Microsoft Backoffice), είτε συνεργάζονται με εταιρίες κατασκευής ΣΔΒΔ και τα προσφέρουν ως ολοκληρωμένα πακέτα με το λειτουργικό τους σύστημα (π.χ., Sun/Oracle).

Ο έλεγχος του υλικού από το λειτουργικό σύστημα, γίνεται σε συνεργασία με λογισμικό που βρίσκεται εγκατεστημένο συνήθως σε μνήμη μόνο ανάγνωσης ή παραλλαγές της (ROM, EPROM κ.λ.π.) και ονομάζεται ενσωματωμένο λογισμικό *(firmware)*. Παραδείγματα ενσωματωμένου λογισμικού αποτελούν το Βασικό Σύστημα Εισόδου – Εξόδου (BIOS) με το οποίο είναι εφοδιασμένοι όλοι οι προσωπικοί υπολογιστές, το λογισμικό των ελεγκτών των σκληρών δίσκων κ.λ.π. Το ενσωματωμένο λογισμικό χρησιμοποιείται επίσης για τον έλεγχο προϊόντων και συστημάτων μαζικής κατανάλωσης και στη βιομηχανία.

Το λογισμικό συστήματος από την μία πλευρά ελέγχει το υλικό, ενώ από την άλλη υποστηρίζει το **λογισμικό εφαρμογών**. Αυτό περιλαμβάνει τις ακόλουθες κατηγορίες:

Επιχειρησιακό λογισμικό. Αποτελεί την πολυπληθέστερη κατηγορία. Συνήθως αποτελείται από ένα σύνολο επιμέρους εφαρμογών (π.χ., μισθοδοσία, διαχείριση αποθήκης, πωλήσεις κ.λ.π.) που ομαδοποιούνται στο λεγόμενο *Πληροφοριακό Σύστημα Διοίκησης (Management Information System – MIS)* και συνήθως αντλούν τα δεδομένα τους από τα Συστήματα Διαχείρισης Βάσεων Δεδομένων. Επιπρόσθετα στην κατηγορία αυτή εντάσσεται και μια σειρά άλλων εφαρμογών, όπως λογισμικό διαχείρισης και προγραμματισμού έργων *(Project Management)*, συντονισμού ομάδων εργασίας, οργάνωσης γραφείου (επεξεργασία κειμένου, λογιστικά φύλλα) κ.λ.π.

Λογισμικό για επιστήμονες και μηχανικούς. Παραδοσιακά σε αυτήν την κατηγορία εντάσσονταν λογισμικό που εκτελούσε πολύπλοκους μαθηματικούς υπολογισμούς σε έναν ευρύτατο τομέα εφαρμογών, από την αστρονομία μέχρι τη μοριακή βιολογία. Σε αυτό προστέθηκε, ιδιαίτερα μετά την ανάπτυξη των γραφικών διεπαφών χρήστη *(Graphical User Interface)*, το λογισμικό σχεδίασης με τη βοήθεια υπολογιστή *(Computer Aided Design – CAD)*, το ιατρικό λογισμικό (κυρίως στη σύνθεση και επεξεργασία ιατρικών εικόνων από τομογράφους, γ-κάμερες κ.λ.π.), καθώς και λογισμικό ανάλυσης και προσομοίωσης (αυτοκινήτων, πλοίων, αεροσκαφών κ.λ.π.).

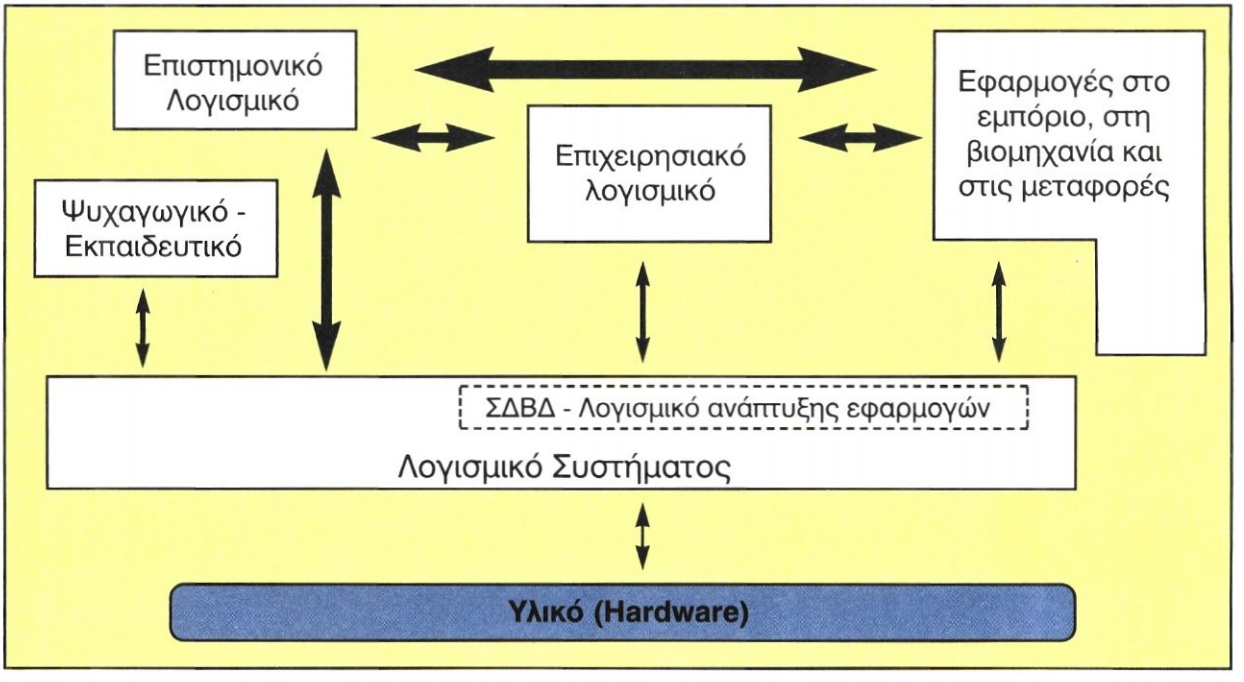
Λογισμικό στο εμπόριο και στη βιομηχανία. Στην κατηγορία αυτή εντάσσονται εφαρμογές λογισμικού που χρησιμοποιούνται στο ηλεκτρονικό εμπόριο (συστήματα κρυπτογραφημένων ηλεκτρονικών συναλλαγών, ανταλλαγής ηλεκτρονικών παραστατικών κ.λ.π.) και στην υποστήριξη της βιομηχανικής παραγωγής από υπολογιστές. Συγκεκριμένα, συστήματα λογισμικού αναλαμβάνουν τον προγραμματισμό και το συντονισμό της παραγωγής και η τεχνική αυτή ονομάζεται *Ολοκληρωμένη Παραγωγή με Υπολογιστή (Computer Integrated Manufacturing – CIM).* Ακόμη υπάρχουν εφαρμογές λογισμικού που χρησιμοποιούνται για τον έλεγχο κοπτικών, τόρνων, αργαλειών, ρομπότ συγκόλλησης κ.λ.π., μια τεχνική που ονομάζεται *Παραγωγή με τη Βοήθεια Υπολογιστή (Computer Aided Manufacturing – CAM).*

Λογισμικό στις συγκοινωνίες και στις μεταφορές. Καλύπτει μια ζώνη μεγάλου εύρους εφαρμογών, από σύνθετες εφαρμογές εγκατεστημένες σε μεγάλους υπολογιστές για τον έλεγχο των σηματοδοτών κυκλοφορίας στις πόλεις, έως ενσωματωμένο λογισμικό στα αυτοκίνητα για έλεγχο του κινητήρα, προβολή χαρτών και αναζήτηση βέλτιστης διαδρομής, εντοπισμό θέσης μέσω δορυφόρου κ.λ.π.

Λογισμικό για εκπαίδευση / ψυχαγωγία. Η κατηγορία αυτή εξαπλώθηκε μαζί με τους προσωπικούς υπολογιστές από τις αρχές της δεκαετίας του '80. Η εξάπλωση της τεχνολογίας των πολυμέσων έδωσε νέα ώθηση στις εφαρμογές αυτές εμπλουτίζοντας σημαντικά και τις δυνατότητες τους. Περιλαμβάνει λογισμικό για επιμόρφωση (εκπαίδευση σε αντικείμενα του σχολείου, κατάρτιση επαγγελματιών, εγκυκλοπαίδειες, αφιερώματα σε μουσεία κ.λ.π.), λογισμικό που ενθαρρύνει τη δημιουργικότητα (σύνθεση μουσικής, ζωγραφική κ.λ.π.), και παιχνίδια.

Οι κατηγορίες λογισμικού που είδαμε μέχρι τώρα νομίζετε ότι είναι ανεξάρτητες ή ότι έχουν κοινά στοιχεία;

Οι κατηγορίες που αναπτύχθηκαν παραπάνω με κανένα τρόπο δεν αποτελούν στεγανά όρια. Το σύγχρονο λογισμικό γίνεται ολοένα και πιο σύνθετο και ορισμένες φορές διαπερνά τα όρια μιας κατηγορίας και αποκτά χαρακτηριστικά λογισμικού άλλης κατηγορίας. Στο **σχήμα 1.1** φαίνεται παραστατικά η συνεργασία διαφόρων κατηγοριών μεταξύ τους. Βλέπουμε για παράδειγμα την εξάρτηση όλων των κατηγοριών λογισμικού από το λογισμικό συστήματος και την ύπαρξη λογισμικού εφαρμογών από όλες τις κατηγορίες (πλην ψυχαγωγικού) που συνεργάζεται με το επιχειρησιακό λογισμικό. Ένα τέτοιο παράδειγμα είναι η συνεργασία του λογισμικού επεξεργασίας ιατρικής εικόνας (επιστημονικό λογισμικό) με το πληροφοριακό σύστημα του νοσοκομείου (επιχειρησιακό λογισμικό). Άλλο παράδειγμα συνεργασίας επιστημονικού με βιομηχανικό λογισμικό είναι η συνεργασία λογισμικού CAD - CIM.



Σχήμα 1.1. Η συνεργασία και αλληλεξάρτηση των κατηγοριών λογισμικού.

Με την εξέλιξη του λογισμικού τα πεδία εφαρμογών του και κατά συνέπεια και οι κατηγορίες του, εμπλουτίζονται ολοένα και περισσότερο

Μπορούν να υπάρξουν και άλλες εναλλακτικές κατηγοριοποιήσεις του λογισμικού, όπως π.χ., ανάλογα με τον αριθμό των χρηστών που υποστηρίζουν (προσωπικό λογισμικό ή λογισμικό πολλών χρηστών), ή το είδος της επικοινωνίας με το χρήστη (γραφικό περιβάλλον, περιβάλλον χαρακτήρων, διακόπτες και φωτοδίοδοι κ.λ.π.).

1.2. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΩΝ ΕΦΑΡΜΟΓΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΔΕΔΟΜΕΝΩΝ

1.2.1. Μια αναδρομή στο λογισμικό διαχείρισης δεδομένων

Αν και οι πρώτοι ηλεκτρονικοί υπολογιστές κατασκευάστηκαν κατά τη διάρκεια του Β' Παγκοσμίου Πολέμου για την επίλυση στρατιωτικών προβλημάτων (π.χ., υπολογισμός βολών, κατασκευή ατομικής βόμβας), εν τούτοις μετά το 1950 επεκτάθηκαν στις επιχειρήσεις, ξεκινώντας από τους μεγάλους κρατικούς οργανισμούς των ΗΠΑ που είχαν τη δυνατότητα να καλύψουν και τα τεράστια κόστη απόκτησης και λειτουργίας μιας τέτοιας μηχανής (βλ. **εικόνα 1.1**).

Εικόνα 1.1. - Ο υπολογιστής 705 της IBM σχεδιασμένος για επιχειρηματικές και λογιστικές εφαρμογές. Μοντέλο του 1954 με λυχνίες, καταλάμβανε μια ολόκληρη αίθουσα. Στη φωτογραφία φαίνονται και οι μονάδες μαγνητικής ταινίας που χρησίμευαν για μόνιμη αποθήκευση δεδομένων.

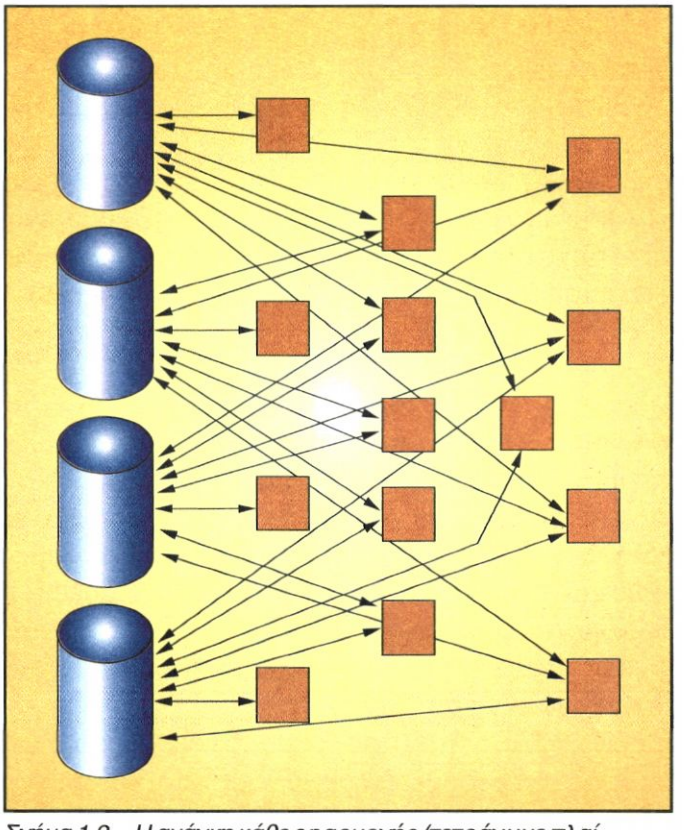
Το λογισμικό που υποστηρίζει την λειτουργία επιχειρήσεων (επιχειρησιακό λογισμικό), όπως είπαμε και στην προηγούμενη παράγραφο, διαχειρίζεται μεγάλο όγκο δεδομένων, τα οποία αρχικά ήταν αποθηκευμένα σε απλά αρχεία, αργότερα οργανώθηκαν σε πιο σύνθετες δομές και διαχειρίζονταν από αυτόνομες εφαρμογές, τα *Συστήματα Διαχείρισης Βάσεων Δεδομένων* (θα μιλήσουμε αναλυτικότερα στην παράγραφο 1.2.3), που εμφανίστηκαν από τα μέσα της δεκαετίας του 1960 και αποτελούν την πλειοψηφία των διαθέσιμων στην αγορά συστημάτων διαχείρισης δεδομένων παγκόσμια.

1.2.2. Δεδομένα οργανωμένα σε αρχεία

Οι πρόγονοι των σημερινών ΣΔΒΔ ήταν προγράμματα διαχείρισης μεγάλων αρχείων δεδομένων. Ο τρόπος οργάνωσης των δεδομένων σε κάθε αρχείο, καθώς και ο τρόπος αποθήκευσης του αρχείου στο φυσικό μέσο (συνήθως μαγνητική ταινία), καθορίζονταν από τους προγραμματιστές κατά τη σχεδίαση και ανάπτυξη του προγράμματος και συνήθως διέφερε από πρόγραμμα σε πρόγραμμα. Κάθε πρόγραμμα επικοινωνούσε με το δικό του αρχείο ή ομάδα αρχείων και ήταν αποκλειστικά αρμόδιο για τη διαχείρισή τους. Οι εφαρμογές αυτές αναπτύσσονταν αρχικά σε συμβολική γλώσσα *(assembly)* και αργότερα (γύρω στο 1960) σε γλώσσες προγραμματισμού όπως π.χ., η COBOL. Τόσο μεγάλη είναι η επένδυση που έχει γίνει σε αυτές τις εφαρμογές, που ακόμα και σήμερα 40 χρόνια μετά χρησιμοποιούνται ακόμα.

Το τμήμα του λογισμικού που είναι αρμόδιο να κωδικοποιήσει στο αρχείο την πληροφορία για λογαριασμό της εφαρμογής, αποτελεί τη φυσική διεπαφή της εφαρμογής προς το αρχείο.

Καθώς κάθε εφαρμογή είναι υπεύθυνη για τη διαχείριση των δικών της αρχείων, αν χρειαζόταν μια εφαρμογή (π.χ., μαθητολόγιο) να χρησιμοποιήσει δεδομένα άλλης (π.χ., αρχείο καθηγητών), έπρεπε να είχε τη δυνατότητα να ανοίξει, να διαβάσει τα αρχεία της και να μπορεί να εντοπίσει μέσα σε αυτά την πληροφορία που θέλει (π.χ., ονοματεπώνυμο καθηγητή). Με βάση και τα προηγούμενα, καταλήγουμε στο συμπέρασμα ότι έπρεπε να ενσωματωθεί στον κώδικά της εφαρμογής μαθητολογίου το τμήμα εκείνο του κώδικα της διαχείρισης καθηγητών που κωδικοποιούσε στο αρχείο τις πληροφορίες, ή όπως ονομάζεται αλλιώς, η **φυσική διεπαφή** για αυτό το αρχείο. Για παράδειγμα, η φυσική διεπαφή καθορίζει με ποια σειρά είναι αποθηκευμένες οι πληροφορίες στο αρχείο, πως αναπαρίστανται σε αυτό (πόσα δυαδικά ψηφία καταλαμβάνει κάθε χαρακτήρας, πόσοι χαρακτήρες δεσμεύονται για την αποθήκευση π.χ., του ονοματεπωνύμου ενός ανθρώπου κ.λ.π.).



Σχήμα 1.2. - Η ανάγκη κάθε εφαρμογής (τετράγωνο πλαίσιο) να διαβάσει δεδομένα από αρχεία (κύλινδροι) άλλων εφαρμογών οδηγεί σε έκρηξη των διεπαφών.

Οι ανάγκες για αμοιβαία επικοινωνία των επιμέρους εφαρμογών οδηγούσαν τα σύνθετα συστήματα σε μια κατάσταση που είναι γνωστή ως έκρηξη των διεπαφών και αναπαρίσταται στο **σχήμα 1.2.** Ακόμη είχαμε το πρόβλημα των **πλεονασματικών δεδομένων**, δηλαδή η ίδια πλη-

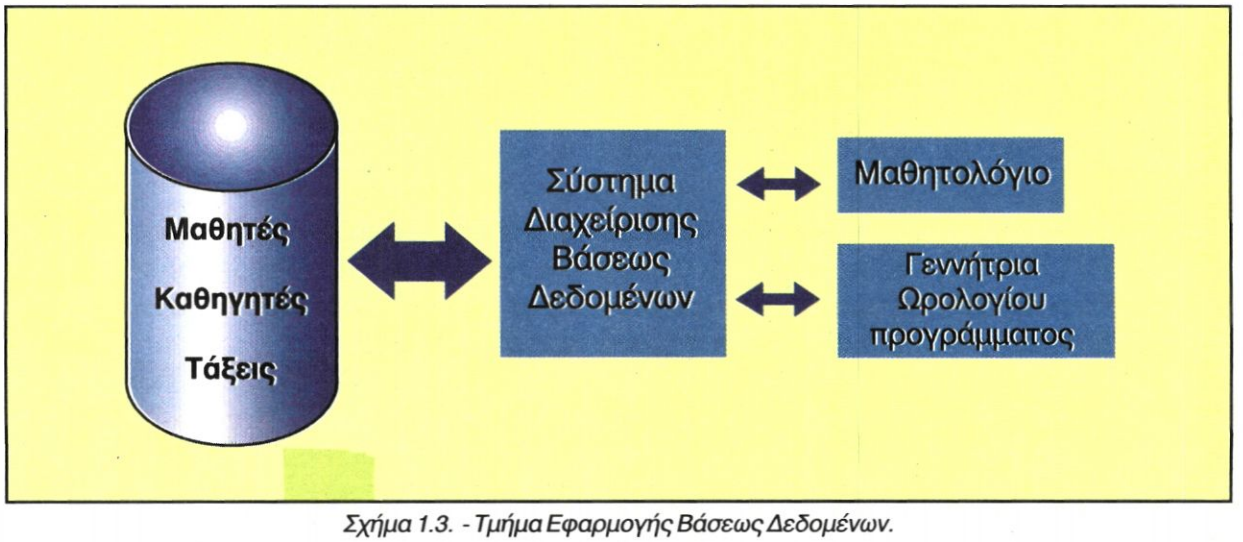
ροφορία να επαναλαμβάνεται σε διαφορετικά αρχεία, καθένα από τα οποία τροφοδοτούσε διαφορετική εφαρμογή. Αυτό δημιουργούσε τις προϋποθέσεις για την εμφάνιση *ασυνεπειών* μεταξύ των διαφόρων αντιγράφων, διότι οι αλλαγές που προκαλούσε μια εφαρμογή στα δικά της δεδομένα δεν συνέβαιναν πάντοτε και στα αντίγραφα αυτών που πιθανά υπήρχαν στα αρχεία των άλλων εφαρμογών. Τέλος, οποιαδήποτε μεταβολή στην οργάνωση και μορφοποίηση των δεδομένων (π.χ., η προσθήκη αριθμού κινητού τηλεφώνου για το αρχείο καθηγητών), απαιτούσε την τροποποίηση του πηγαίου κώδικα των εφαρμογών και την επαναμεταγλώττιση τους, καθώς και την μετατροπή των παλιών αρχείων στη νέα μορφή.

Περισσότερα για την οργάνωση των δεδομένων σε αρχεία, θα ξαναδούμε στο κεφάλαιο 6.

1.2.3. Συστήματα Διαχείρισης Βάσεων Δεδομένων

Σε μια προσπάθεια να ξεπεραστούν τα προβλήματα των εφαρμογών διαχείρισης αρχείων, η ευθύνη διαχείρισης των δεδομένων και οργάνωσής τους σε αρχεία αποσπάστηκε από τις εφαρμογές. Αντί τα δεδομένα να βρίσκονται επιμερισμένα σε διαφορετικές εφαρμογές μορφοποιημένα σύμφωνα με τα χαρακτηριστικά της κάθε μιας, φιλοξενούνται από μια σύνθετη συλλογή με συγκεκριμένη οργάνωση, που ονομάζεται **βάση δεδομένων**. Ανάλογα με την υλοποίησή της, η βάση δεδομένων μπορεί να είναι αποθηκευμένη σε ένα αρχείο ή μία ομάδα αρχείων. Η απλοποιημένη αρχιτεκτονική ενός τμήματος μιας βάσεως δεδομένων που υποστηρίζει μια εφαρμογή διαχείρισης σχολείου απεικονίζεται στο **σχήμα 1.3**.

Βάση δεδομένων είναι μία συλλογή δεδομένων οργανωμένη με σαφήνεια



Σχήμα 1.3. - Τμήμα Εφαρμογής Βάσεως Δεδομένων.

Ο τρόπος οργάνωσης των δεδομένων μέσα στο αρχείο και γενικά η διαχείρισή τους (προσθήκη, διαγραφή, ενημέρωση, αναζήτηση κ.λ.π.), γίνεται από ομάδα εφαρμογών λογισμικού, που λέγεται **Σύστημα Διαχείρισης Βάσεως Δεδομένων ΣΔΒΔ)**. Μερικές από τις λειτουργίες που επιτελεί ένα ΣΔΒΔ είναι οι ακόλουθες:

- Αναλαμβάνει να οργανώσει τη βάση δεδομένων στο μέσο αποθήκευσης (συνήθως σκληροί δίσκοι).
- Διαθέτει τους μηχανισμούς για τη διαχείριση των δεδομένων.
- Αναλαμβάνει να τροφοδοτήσει τις εφαρμογές με δεδομένα στην μορφή που αυτές τα ζητούν.

Στη συνέχεια του κεφαλαίου (και αναλυτικότερα στο κεφάλαιο 6) θα εξετάσουμε και άλλες λειτουργίες των ΣΔΒΔ.

Το ΣΔΒΔ αναλαμβάνει να οργανώσει τα δεδομένα, να τα αποθηκεύσει στο φυσικό μέσο και να τροφοδοτήσει τις εφαρμογές μέσω μιας κοινής διεπαφής.

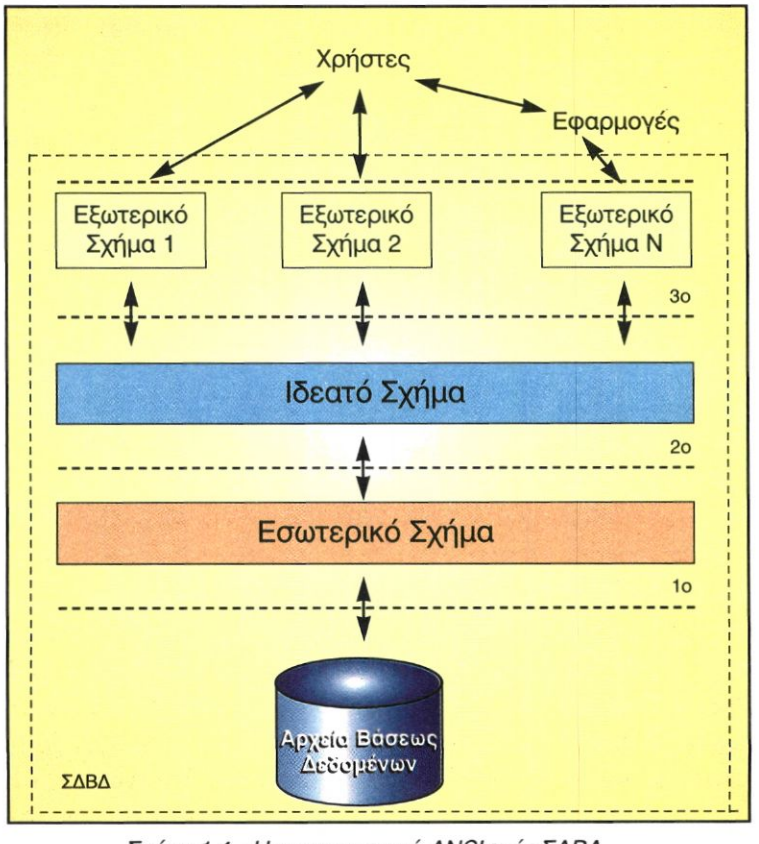
Από τη στιγμή που οι εφαρμογές έρχονται σε επαφή με το ΣΔΒΔ και όχι με τα επιμέρους δεδομένα ξεχωριστά, τότε χρειάζονται μόνο μια διεπαφή. Η κοινή αυτή διεπαφή επιτρέπει την προσθήκη και άλλων εφαρμογών, χωρίς να υπάρχει ο κίνδυνος της έκρηξης των διεπαφών, που είδαμε στην προηγούμενη παράγραφο. Επιπρόσθετα, αυτή η επικοινωνία δε χρειάζεται να καλύψει θέματα σχετικά με το πώς είναι τα δεδομένα αποθηκευμένα στον υπολογιστή, πόσα bytes καταλαμβάνουν κ.λ.π., παρά φροντίζει να παράσχει στην εφαρμογή τα δεδομένα που ζητά στη μορφή που τα ζητά. Αυτό με άλλα λόγια λέγεται **λογική διεπαφή.**

Αυτή η ιδιότητα, δηλαδή η επικοινωνία να γίνεται μέσω λογικής διεπαφής, χωρίς οι εφαρμογές να ασχολούνται με την αποθήκευση στο φυσικό μέσο (δίσκος, ταινία, CDROM κ.λ.π.), αναφέρεται ως **ανεξαρτησία δεδομένων.** Σε αυτήν κυρίως οφείλεται η ικανότητα των ΣΔΒΔ να μοιράζουν τα δεδομένα μεταξύ ξεχωριστών εφαρμογών. Η ανεξαρτησία δεδομένων προσφέρει ένα ακόμη πλεονέκτημα. Αν χρειαστεί να τροποποιηθεί η οργάνωση των αρχείων της βάσης δεδομένων, οι εφαρμογές δεν ενοχλούνται καθόλου στο βαθμό που η λογική διεπαφή δεν μεταβάλλεται. Απλά, το σύστημα διαχείρισης βάσεως δεδομένων αναλαμβάνει να αντιστοιχήσει τις αιτήσεις για δεδομένα των εφαρμογών στη νέα οργάνωση. Επίσης και οι εφαρμογές μπορούν να αναβαθμιστούν ανεξάρτητα, χωρίς να πειράξουμε τη βάση δεδομένων.

Το 1978 η Αμερικανική Επιτροπή Εθνικών Προτύπων *(American National Standards Committee - ANSI)* προσδιόρισε μια πρότυπη αρχιτεκτονική τριών επιπέδων για τα ΣΔΒΔ. Τα επίπεδα αυτά ή **Σχήματα** (βλ. **σχήμα 1.4**) περιγράφουν τους διαφορετικούς τρόπους αναπαράστασης των δεδομένων σε ένα ΣΔΒΔ. Κάθε ένα από αυτά τα Σχήματα είναι ανεξάρτητο από τα άλλα και δεν επηρεάζεται από ενδεχόμενες αλλαγές στο εσωτερικό τους.

Το **Εσωτερικό Σχήμα** είναι η αναπαράσταση των δεδομένων σε μορφή κατάλληλη για να αποθηκευθούν στο φυσικό μέσο (π.χ., σκληρός δίσκος, μνήμη RAM, κ.λ.π.).

Το **Ιδεατό Σχήμα** *(conceptual schema)* αναπαριστά τις έννοιες που αντιπροσωπεύονται από το σύνολο των αποθηκευμένων δεδομένων. Η αναπαράσταση αυτή δηλώνει ουσιαστικά τη λογική δομή της βάσης δεδομένων, δηλαδή το τι αντιπροσωπεύει η πληροφορία που είναι αποθηκευμένη στη βάση, ποιες ιδιότητες έχει και πώς ονομάζονται, τι τιμή έχουν και ποια είναι η μεταξύ τους σχέση.



Σχήμα 1.4. - Η αρχιτεκτονική ANSI ενός ΣΔΒΔ.

Το τρίτο επίπεδο συντίθεται από τα **Εξωτερικά Σχήματα**, που αναπαριστούν τα δεδομένα στη μορφή που επιθυμούν οι χρήστες ή οι εφαρμογές που επικοινωνούν με το ΣΔΒΔ. Τα δεδομένα αυτά συνήθως είναι υποσύνολο του συνόλου των δεδομένων που βρίσκονται αποθηκευμένα στη βάση δεδομένων, ή παράγωγα επεξεργασίας αυτών (π.χ., υπολογισμός βαθμολογίας απολυτηρίου κ.λ.π.). Συνήθως τα σχήματα αυτά είναι περισσότερα από ένα, ανάλογα με τις απαιτήσεις ή τα δικαιώματα που έχει κάθε ομάδα χρηστών.

Η αναπαράσταση των αποθηκευμένων δεδομένων στο ΣΔΒΔ σε τρία διακριτά και ανεξάρτητα μεταξύ τους Σχήματα εξασφαλίζει την ανεξαρτησία των δεδομένων για την οποία μιλήσαμε προηγουμένως.

Η κεντρική διαχείριση όλων των δεδομένων για λογαριασμό διαφόρων εφαρμογών και ομάδων χρηστών, επιφορτίζει τα ΣΔΒΔ με πρόσθετα καθήκοντα. Πρέπει να προσφέρουν διάφορα επίπεδα διαβαθμίσεων ασφάλειας, έτσι ώστε ο μη εξουσιοδοτημένος χρήστης να εμποδίζεται από το να δει ή να αλλάξει δεδομένα στα οποία δεν πρέπει να έχει πρόσβαση. Επίσης πρέπει να διαθέτουν ειδικούς μηχανισμούς ελέγχου ταυτόχρονης προσπέλασης (concurrency control), που σημαίνει ότι οι ενδεχομένως ταυτόχρονες εργασίες που εκτελούν διαφορετικοί χρήστες θα οργανώνονται αυτόματα από το ΣΔΒΔ με τέτοιον τρόπο, ώστε να μην προκαλεί προβλήμα-

τα η μια στην εκτέλεση της άλλης. Ακόμα τα ΣΔΒΔ πρέπει να είναι εφοδιασμένα με μηχανισμούς ανάκτησης *(recovery)* των δεδομένων σε περίπτωση που το σύστημα πάθει βλάβη ξαφνικά από κάποια αιτία (π.χ., διακοπή ρεύματος).

Τα ΣΔΒΔ δεν παρουσιάζουν μόνο πλεονεκτήματα. Τα μειονεκτήματα τους είναι αφενός το κόστος σε λογισμικό και ενδεχομένως πρόσθετο υλικό που απαιτούν και αφετέρου το κόστος λειτουργίας τους (τα μεγάλα ΣΔΒΔ συνήθως απαιτούν και έναν εξειδικευμένο διαχειριστή). Βέβαια αυτά είναι μικρά και συνήθως μπορούν να αντισταθμιστούν από την αύξηση στην παραγωγικότητα που επιτυγχάνουν.

1.2.4. Μοντέλα Δεδομένων

Όπως είδαμε και στην ενότητα 1.2.3, το ιδεατό σχήμα εμπεριέχει τη λογική δομή της βάσης δεδομένων, ή με άλλα λόγια ένα **μοντέλο** του πραγματικού συστήματος (π.χ., σχολείο, επιχείρηση κ.λ.π.). Σημείο κλειδί λοιπόν στη σχεδίαση και υλοποίηση μιας βάσης δεδομένων είναι η διαδικασία σχηματισμού αυτού του μοντέλου της πραγματικότητας.

Παράδειγμα: Για να σχηματίσουμε ένα μοντέλο, όχι μόνο στην πληροφορική αλλά και σε κάθε μας δραστηριότητα, ακολουθούμε κάποια βήματα που είναι λίγο πολύ κοινά. Ας ρίξουμε μια ματιά στο **σχήμα 1.5** που απεικονίζει τη σχέση του πλανήτη Γη με το μοντέλο του (μια υδρόγειο σφαίρα):

1. Το πρώτο βήμα που κάνουμε για την κατασκευή του μοντέλου μας είναι να αποφασίσουμε ποια από τα χαρακτηριστικά και τις λειτουργίες του πραγματικού αντικειμένου θα συμπεριλάβουμε. Κάθε μοντέλο είναι μια αφαίρεση της πραγματικότητας, δηλαδή ενσωματώνει μόνο αυτά που μας ενδιαφέρουν, τα υπόλοιπα τα παραβλέπουμε.
2. Αφού λοιπόν αποφασίσουμε, θα αναλύσουμε αυτήν την πληροφορία για να προσδιορίσουμε τι δομές θα σχηματίσουμε στο μοντέλο μας. Για παράδειγμα, θα απεικονίσουμε τα κράτη ή απλά θα δείχνουμε τις ηπείρους; Έχοντας απαντήσει και σε αυτά τα ερωτήματα θα έχουμε σχηματίσει τις αφηρημένες, ή αλλιώς *ιδεατές* προδιαγραφές του μοντέλου μας.
3. Στη συνέχεια, θα αποφασίσουμε με ποιο τρόπο θα αναπαραστήσουμε τις δομές (ηπείρους, κράτη, ποτάμια, θάλασσες κ.λ.π.) του μοντέλου μας.
4. Μόλις ολοκληρώσουμε και αυτό το βήμα θα πρέπει να ελέγξουμε το μοντέλο μας αν ικανοποιεί τις απαιτήσεις που έχουμε από αυτό (αν κάνει τη δουλειά για την οποία σχεδιάστηκε).
5. Ακόμα όμως και μετά απ' αυτόν τον έλεγχο δεν έχουμε ολοκληρώσει το μοντέλο μας. Πρέπει τώρα να φτιάξουμε αναλυτικές οδηγίες προς τους τεχνίτες που θα φτιάξουν την υδρόγειο (π.χ., ξυλουργός, γραφίστας κ.λ.π.).
6. Τέλος, μόλις ολοκληρωθεί και η φυσική κατασκευή του μοντέλου μας και μας παραδοθεί η υδρόγειος, πρέπει να την ελέγξουμε για μια ακόμη φορά για να βεβαιωθούμε ότι όχι μόνο κάνει αυτό για το οποίο σχεδιάστηκε, αλλά και ότι το κάνει καλά.

Σχήμα 1.5. - Η διαδικασία κατασκευής ενός προτύπου.

Ανάλογα είναι και τα βήματα που κάνουμε για να σχεδιάσουμε και υλοποιήσουμε μια βάση δεδομένων. Αναλυτικότερα θα αναφερθούμε σε αυτά στις επόμενες ενότητες. Προς το παρόν παραθέτουμε στον **πίνακα 1.1** τα βήματα αυτά σε απ' ευθείας αντιστοιχία με τα βήματα που αναφέραμε προηγουμένως.

Βήμα	Περιγραφή
1	Ανάλυση απαιτήσεων[Τι θέλουμε να προτυποποιήσουμε; (μοντελοποιήσουμε)].
2	Δημιουργία Ιδεατού Προτύπου (μοντέλου) (Ποιες πληροφορίες θα χρειαστούμε να συμπεριλάβουμε στο πρότυπό μας;).
3	Σχεδίαση Ιδεατού Σχήματος (Πώς μπορούμε να εκφράσουμε αυτές τις πληροφορίες;).
4	Λειτουργική Ανάλυση (Μας επιτρέπει αυτή η έκφραση να κάνουμε ότι χρειαζόμαστε;).
5	Καθορισμός Σχήματος - Φυσική Σχεδίαση (Πώς μπορούμε να υλοποιήσουμε αυτήν την έκφραση στο μέσο του προτύπου μας;).
6	Ανάλυση Επιδόσεων (Ικανοποιεί το πρότυπό μας τους σκοπούς για τους οποίους κατασκευάστηκε;).

Πίνακας 1.1. - Βήματα για τη σχεδίαση και υλοποίηση μιας βάσης δεδομένων.

Το σκεπτικό πίσω από τη δημιουργία του ιδεατού μοντέλου είναι ότι τα δεδομένα μας αφορούν *οντότητες*.

Ορισμός **Οντότητα** είναι κάθε συστατικό του υπό εξέταση πραγματικού συστήματος που έχει αυτόνομη ύπαρξη και για το οποίο μας ενδιαφέρει να συλλέξουμε πληροφορίες.

Κάθε οντότητα έχει κάποιες **ιδιότητες** *(attributes)* και μπορεί με κάποιο τρόπο να **συσχετιστεί** με άλλες οντότητες. Αυτές οι οντότητες μπορεί να είναι πρόσωπα, πράγματα, γεγονότα (όπως π.χ., η πώληση ενός προϊόντος), ή δραστηριότητες (π.χ., η παραγωγή ενός προϊόντος). Οι ιδιό-

τητες αυτές διαφοροποιούν το κάθε **στιγμιότυπο** (δηλαδή τα δεδομένα ενός συγκεκριμένου προσώπου, πράγματος, γεγονότος ή δραστηριότητας) μιας οντότητας από άλλα της ίδιας οντότητας. Για παράδειγμα το ονοματεπώνυμο, ημερομηνία γέννησης, φύλλο κ.λ.π. ενός μαθητή μας βοηθούν να τον ξεχωρίσουμε από τους συμμαθητές του. Οι συσχετίσεις σχηματίζονται μεταξύ δύο ή περισσότερων οντοτήτων (για παράδειγμα ή εγγραφή ενός μαθητή σε μία τάξη, η πρόσληψη ενός υπαλλήλου, ο δανεισμός βιβλίων από αναγνώστες σε μια δανειστική βιβλιοθήκη κ.λ.π.). Τις οντότητες θα τις εξετάσουμε αναλυτικότερα στο κεφάλαιο 6.

Στο ιδεατό πρότυπο προσδιορίζουμε τις οντότητες και στο ιδεατό σχήμα προσδιορίζουμε μια τεχνική αναπαράστασής τους.

Κατά τη φάση της σχεδίασης του ιδεατού σχήματος, οργανώνουμε το ιδεατό μοντέλο σε τέτοια μορφή έτσι ώστε να μπορεί να αναπαρασταθεί από το κατάλληλο ΣΔΒΔ. Κάθε ΣΔΒΔ χρησιμοποιεί ένα συγκεκριμένο είδος δομών δεδομένων (π.χ., ιεραρχικά δένδρα, πίνακες) για να αναπαραστήσει τις οντότητες και τις συσχετίσεις του ιδεατού μοντέλου.

Ορισμός

Το είδος των δομών δεδομένων που χρησιμοποιεί το κάθε ΣΔΒΔ ως δομικό λίθο για την κατασκευή του ιδεατού σχήματος αναφέρεται και σαν **μοντέλο δεδομένων** και ανάλογα με αυτό τα ΣΔΒΔ χωρίζονται σε *ιεραρχικά, δικτυωτά, σχεσιακά.*

Στις επόμενες παραγράφους θα παρουσιάσουμε εν συντομία τα τρία μοντέλα. Το σχεσιακό, στο οποίο βασίζονται και τα περισσότερα διαθέσιμα ΣΔΒΔ στην αγορά αυτή τη στιγμή, θα το εξετάσουμε αναλυτικότερα στο κεφάλαιο 6. Πρόσφατα έχουν εμφανιστεί και συστήματα δεδομένων που είτε εμπλουτίζουν το σχεσιακό μοντέλο με αντικειμενοστρεφείς (object-oriented) δυνατότητες, είτε εφαρμόζουν ένα πλήρως αντικειμενοστρεφές μοντέλο οργάνωσης των δεδομένων. Για αυτά θα κάνουμε μια σύντομη αναφορά στο κεφάλαιο 3, αλλά δε θα μας απασχολήσουν αναλυτικότερα σε αυτό το βιβλίο.

1.2.5.ΣχεσιακάΣυστήματα Διαχείρισης Βάσεων Δεδομένων

Οι βάσεις για το σχεσιακό μοντέλο (και για τα σχεσιακά ΣΔΒΔ που το χρησιμοποιούν), τέθηκαν από τον E. F. Codd με μια κλασσική εργασία του, που δημοσιεύθηκε στο τεχνικό περιοδικό Communications of the Association of Computing Machinery τον Ιούνιο του 1970. Κατά την περίοδο 1970-4 με αλλεπάλληλες εργασίες ολοκληρώνει την τεκμηρίωση του μοντέλου και μέχρι το 1985 τα σχεσιακά συστήματα είχαν επικρατήσει στην αγορά των ΣΔΒΔ.

Το σχεσιακό μοντέλο βασίζεται στην απλούστερη των δομών δεδομένων, τον **πίνακα**. Κάθε **στήλη** του πίνακα αντιστοιχεί σε μία **ιδιότητα** (μιας οντότητας ή μιας συσχέτισης) και το σύνολο των τιμών των στηλών σε κάθε γραμμή του πίνακα αντιστοιχεί σε ένα στιγμιότυπο μιας οντότητας ή μιας συσχέτισης. Ο πίνακας αναπαριστά μια **σχέση** μεταξύ των ιδιοτήτων (στηλών) του και γι' αυτόν τον λόγο οι πίνακες ονομάζονται και *σχέσεις.*

Όπως είπαμε ο πίνακας χρησιμοποιείται για την αναπαράσταση τόσο των οντοτήτων, όσο και των συσχετίσεων. Στο σχεσιακό μοντέλο δε χρειάζονται πρόσθετες δομές. Ακόμα και σύνθετες δομές, όπως οι σχέσεις πολλά προς πολλά στο σχεσιακό μοντέλο αναπαρίστανται με πίνακες.

Στο παράδειγμα της βάσης δεδομένων του σχολείου, για την οντότητα Μάθημα μπορεί να έχουν προσδιοριστεί κατά το σχηματισμό του ιδεατού μοντέλου οι ιδιότητες *κωδικός (μαθήματος), τίτλος και τάξη*. Για την οντότητα αυτή μπορούμε να κατασκευάσουμε μια σχέση απλά ορίζοντας έναν πίνακα με τρεις στήλες και ονομάζοντας κάθε μία απ' αυτές με το αντίστοιχο όνομα της ιδιότητας της οντότητας (βλ. **πίνακα 1.2**).

Η συντριπτική πλειοψηφία των εφαρμογών διαχείρισης βάσεων δεδομένων για προσωπικούς υπολογιστές χρησιμοποιεί τον πίνακα για την κατασκευή του ιδεατού σχήματος. Μήπως μπορείτε να δώσετε παραδείγματα;

Κωδικός	Τίτλος	Τάξη
101	Μαθηματικά	Α'Λυκείου
204	Έκθεση	Β' Λυκείου
105	Ιστορία	Α' Λυκείου

Πίνακας 1.2. - Μία απλή σχέση για την οντότητα "Μάθημα".

Η σειρά με την οποία παραθέτονται οι στήλες σε έναν πίνακα δεν έχει σημασία. Δηλαδή η πληροφορία που περιέχεται σε μία γραμμή του πίνακα δεν εξαρτάται από τη διάταξη των ιδιοτήτων της. Μπορούμε ελεύθερα να αντιμεταθέσουμε τη σειρά εμφάνισης των στηλών του πίνακα 1.2 (αντί για κωδικός – τίτλος – τάξη να γίνει τίτλος – τάξη – κωδικός). Η κάθε γραμμή του πίνακα ονομάζεται και **πλειάδα** *(tuple)*.

Αντίστοιχα με την ανεξαρτησία της πληροφορίας από τη διάταξη των στηλών, ούτε η διάταξη των πλειάδων (γραμμών) επηρεάζει την πληροφορία που είναι αποθηκευμένη σε έναν πίνακα.

Η στήλη ονομάζεται και **πεδίο**. Με αυτό υποδηλώνουμε ότι κάθε στιγμιότυπο μιας ιδιότητας μπορεί να πάρει τιμές από ένα σύνολο δυνατών τιμών, το πεδίο τιμών του. Αυτό προσδιορίζεται από τον τύπο δεδομένων της ιδιότητας (χαρακτήρας, αριθμός κ.λ.π.) και από περιορισμούς που απορρέουν από τη φυσική σημασία της (π.χ., δεν μπορούμε να έχουμε αρνητικές ηλικίες). Η ονομασία αυτή θέλει να δώσει έμφαση στο γεγονός ότι ο πίνακας δεν αναπαριστά τόσο τις τιμές των ιδιοτήτων, όσο την αντιστοιχία μεταξύ των τιμών τους. Έτσι μπορούμε να φανταστούμε την πλειάδα όχι σαν μια σειρά από υποδοχές που στην κάθε μια μπορούμε να βάζουμε αυθαίρετα όποια τιμή θέλουμε, αλλά σαν μια επιλογή μιας τιμής από κάθε πεδίο, για να περιγράψει το στιγμιότυπο μιας οντότητας.

Λίγο μετά την εμφάνιση του σχεσιακού μοντέλου, παρουσιάστηκε μια γλώσσα από την IBM που γρήγορα έγινε το διεθνές πρότυπο για την επικοινωνία του χρήστη και του προγραμματιστή με όλα τα σχεσιακά ΣΔΒΔ. Η γλώσσα αυτή λέγεται *SQL (Structured Query Language – Δομημένη Γλώσσα Ερωτήσεων)* και σε σχέση με τις μέχρι τότε γλώσσες παρουσίαζε τα πλεονεκτήματα της γρήγορης εκμάθησης και της μικρότερης απόστασης από την καθομιλούμενη γλώσσα των ανθρώπων.

Η SQL έχει καθιερωθεί σαν πρότυπο για την επικοινωνία και τη διαχείριση των σχεσιακών ΣΔΒΔ.

Για παράδειγμα στα αγγλικά λέμε:

Παράδειγμα:

SELECT the fields labeled "code", "name" and "price" FROM the "Stock" table and only show entries having a stored "price" value which is GREATER THAN 5000; present the results ORDERED alphabetically by the contents of the "name" field

Που στα ελληνικά μπορούμε να το αποδώσουμε ως:
ΕΠΙΛΕΞΕ τα πεδία "κωδικός", "όνομα" and "τιμή" ΑΠΟ τον πίνακα "Εμπορεύματα" και δείξε μόνο τις καταχωρήσεις που έχουν τιμή αποθηκευμένη στο πεδίο "τιμή" ΜΕΓΑΛΥΤΕΡΗ ΑΠΟ 5000, παρουσίασε τα αποτελέσματα ΤΑΞΙΝΟΜΗΜΕΝΑ αλφαβητικά σύμφωνα με τα περιεχόμενα του πεδίου "όνομα"

Η αντίστοιχη σύνταξη σε SQL είναι:

SELECT code, name, price
FROM Stock
WHERE price > 5000
ORDER BY name

Η SQL είναι γλώσσα δηλωτική *(declarative)* και όχι διαδικαστική *(procedural)* όπως είναι οι περισσότερες από τις γνωστές γλώσσες προγραμματισμού (π.χ., Basic, Pascal, C κ.λ.π.). Πέρα από τις εντολές για τον ορισμό των δεδομένων (π.χ., δημιουργία πινάκων και ορισμό πεδίων) και τη διαχείρισή τους (π.χ., εισαγωγή, διαγραφή, αναζήτηση δεδομένων) δεν περιλαμβάνει εντολές για ορισμό συναρτήσεων ή διαδικασιών, δομές ελέγχου (π.χ., if then else...) και επανάληψης (π.χ., while do...). Οι εντολές της SQL ομαδοποιούνται σε δύο ομάδες, τη *Γλώσσα Ορισμού Δεδομένων (D.D.L. – Data Definition Language)* και τη *Γλώσσα Διαχείρισης Δεδομένων (D.M.L. – Data Manipulation Language)*. Ο διαχωρισμός αυτός εμφανίστηκε για πρώτη φορά στη γλώσσα διαχείρισης δικτυωτών ΣΔΒΔ και διατηρήθηκε αργότερα και στην SQL.

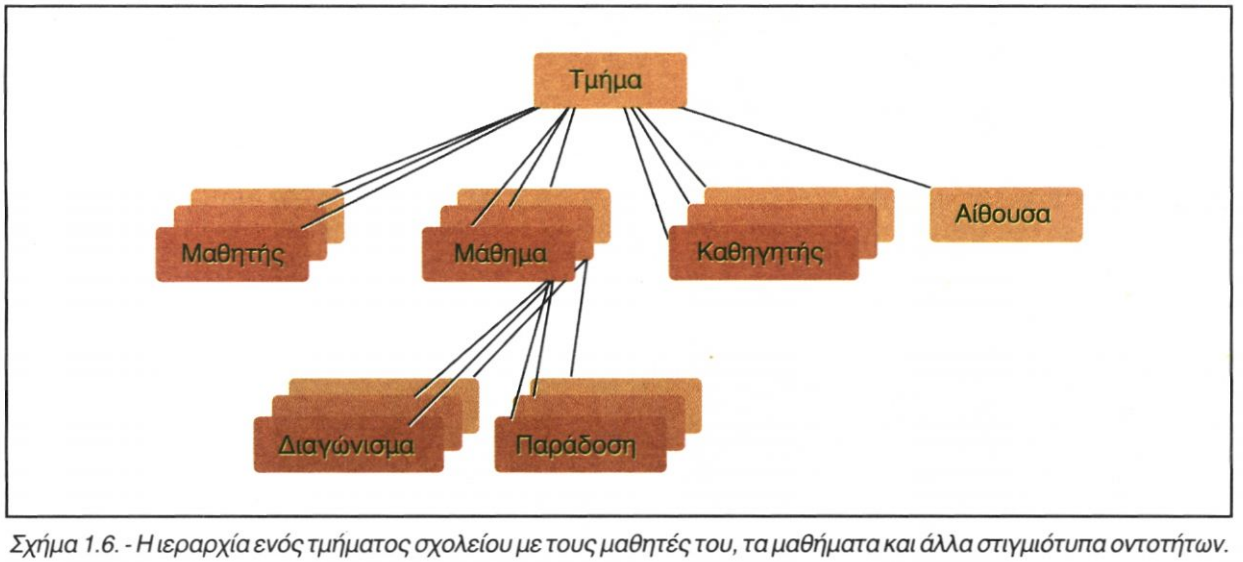
Τα σχεσιακά ΣΔΒΔ υπερτερούν αισθητά σε σχέση με τα ιεραρχικά και δικτυωτά στην ευκολία διαχείρισης και συντήρησης των δεδομένων, στην ευκολία χρήσης και διαθέτουν μια πληθώρα εργαλείων ταχείας ανάπτυξης εφαρμογών για διάφορα συστήματα. Τα διαθέσιμα σήμερα στην αγορά ΣΔΒΔ είναι στη συντριπτική πλειοψηφία τους σχεσιακά, όμως μεγάλη ποσότητα επιχειρησιακών (π.χ., τράπεζες, ασφαλιστικοί οργανισμοί κ.λ.π.) δεδομένων εξακολουθεί να βρίσκεται σε συστήματα διαχείρισης αρχείων, ιεραρχικά και δικτυωτά ΣΔΒΔ. Καθότι όμως δε γίνονται νέες εγκαταστάσεις τέτοιων ΣΔΒΔ το ποσοστό τους διαρκώς μειώνεται.

Παραδείγματα σχεσιακών ΣΔΒΔ είναι τα Oracle, Ingres, Unify, Sybase, Informix, DB2 της IBM, αλλά και μικρότερα συστήματα (ή προσωπικά ΣΔΒΔ), που λειτουργούν σε προσωπικούς υπολογιστές όπως η Microsoft Access, η dBase IV κ.λ.π. Η διαφορές των πρώτων προϊόντων από τα προσωπικά είναι οι ακόλουθες:

- Τα μεγάλα ΣΔΒΔ διαθέτουν εξελιγμένους μηχανισμούς ελέγχου ταυτόχρονης προσπέλασης και ανάκτησης δεδομένων (βλ. ενότητα 1.2.3.).
- Τα περισσότερα σύγχρονα μεγάλα ΣΔΒΔ λειτουργούν σύμφωνα με το πρότυπο πελάτη – εξυπηρετητή (θα το δούμε αναλυτικότερα στο επόμενο κεφάλαιο).
- Τα μεγάλα ΣΔΒΔ διαθέτουν πολύπλοκους βελτιστοποιητές που αναλαμβάνουν να εκτελέσουν τις ερωτήσεις στη βάση δεδομένων κατά τον ταχύτερο δυνατό τρόπο.
- Τα μεγάλα ΣΔΒΔ υποστηρίζουν καλύτερα την εργασία πολλών χρηστών και προσφέρουν πλήρεις ρυθμίσεις δικαιωμάτων του κάθε χρήστη σε αυτά (π.χ., ποιος είναι ο ιδιοκτήτης κάποιου πίνακα, ποιοι μπορούν να τον διαβάσουν, να γράψουν σε αυτόν, να σβήσουν δεδομένα κ.λ.π.).
- Τα προσωπικά ΣΔΒΔ προσφέρουν συνήθως σε ενιαίο περιβάλλον εργαλεία σχεδίασης και ανάπτυξης εφαρμογών (μέχρι και σχεδίαση φορμών και μενού), ενώ τα μεγάλα προσφέρουν μια πληθώρα επιμέρους εφαρμογών για τη διαχείριση και την προστασία των δεδομένων, και όχι τόσο για τη σχεδίαση και ανάπτυξη εφαρμογών (αυτά προσφέρονται ξεχωριστά, είτε από τις εταιρίες κατασκευής τους, είτε από τρίτους κατασκευαστές). Ανάμεσα στις επιμέρους εφαρμογές που συνοδεύουν τα μεγάλα ΣΔΒΔ είναι η εφαρμογή διαχείρισης της βάσης, εφαρμογές ορισμού και διαχείρισης χρηστών, εφαρμογές δημιουργίας εφεδρικών αντιγράφων, εφαρμογές παρακολούθησης της απόδοσης του συστήματος, κονσόλα για διαχείριση μέσω SQL κ.λ.π.

1.2.6. ΙεραρχικάΣυστήματα Διαχείρισης Βάσεων Δεδομένων

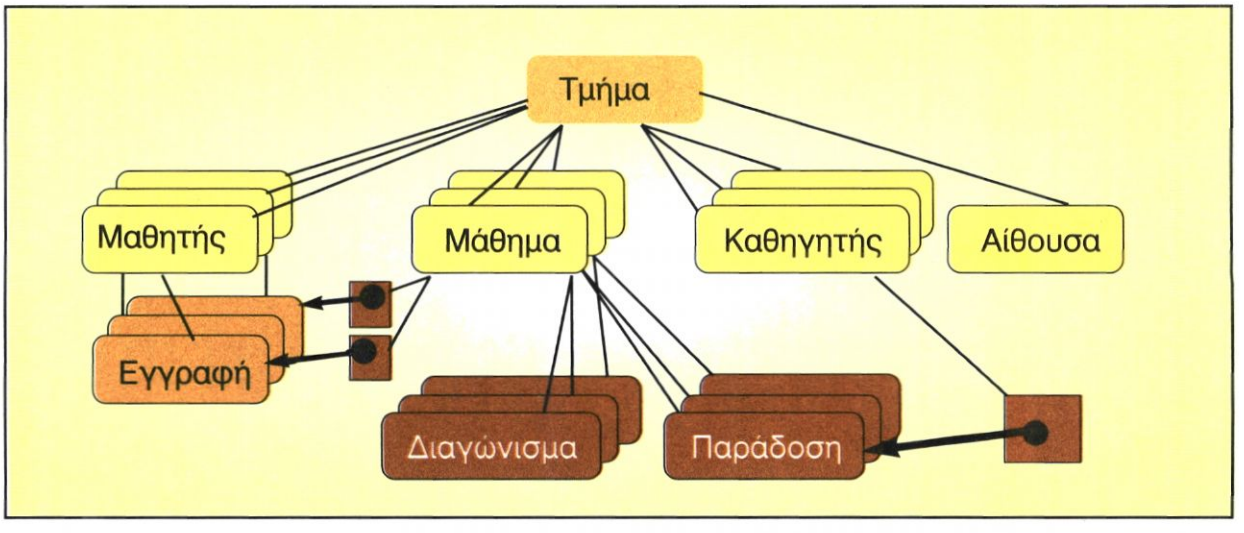
Πρόκειται για τα πρώτα χρονολογικά ΣΔΒΔ και βασίζονται στο ιεραρχικό μοντέλο δεδομένων. Κυριάρχησαν στην αγορά από τα μέσα της δεκαετίας του '60 και σχεδόν σε ολόκληρη τη δεκαετία του '70. Διασημότερος εκπρόσωπός τους είναι το σύστημα IMS της IBM, που εμφανίστηκε το 1968, ενώ μεταξύ άλλων μερίδιο της αγοράς εκείνη την εποχή πήραν και τα TDMS, MARS-VI και System-2000. Στο ιεραρχικό μοντέλο οι οντότητες αναπαριστάνονται σαν οι κόμβοι ενός δένδρου και συσχετίζονται μεταξύ τους με σχέση ένα προς ένα ή ένα προς πολλά. Οι συσχετίσεις αυτές αποτε-



Σχήμα 1.6. - Η ιεραρχία ενός τμήματος σχολείου με τους μαθητές του, τα μαθήματα και άλλα στιγμιότυπα οντοτήτων.

λούν τα κλαδιά του δένδρου. Στο **σχήμα 1.6** απεικονίζεται το στιγμιότυπο ενός τμήματος σε ένα σχολείο μαζί με στιγμιότυπα μαθητών που ανήκουν σε αυτό, μαθημάτων κ.λ.π.

Όπως φαίνεται και στο σχήμα 1.6, μία οντότητα ξεχωρίζει διότι σε όλες τις συχετίσεις της με τις άλλες, είτε είναι αυτές της μορφής ένα προς ένα, είτε ένα προς πολλά, είναι πάντα από την μεριά του "ένα". Αυτή η οντότητα (στην εικόνα μας το "Τμήμα"), λέγεται *ρίζα*.



Σχήμα 1.7. - Η ιεραρχία συμπληρωμένη

Οι οντότητες που δεν συσχετίζονται παρακάτω με άλλες οντότητες ονομάζονται *φύλλα* (στο παράδειγμα μας φύλλα είναι το "Διαγώνισμα", η "Παράδοση" και η "Αίθουσα").

Τι γίνεται όμως όταν μια οντότητα μετέχει σε δύο διαφορετικές συσχετίσεις; Τέτοια είναι η οντότητα "Παράδοση" του παραδείγματός μας που μετέχει και στη συσχέτιση με το "Μάθημα", αλλά και στη συσχέτιση με τον "Καθηγητή", η οποία δεν απεικονίζεται στο σχήμα 1.6. Επίσης πώς αναπαριστούμε τις συσχετίσεις πολλά προς πολλά, όπως είναι η σχέση μεταξύ μαθητών και μαθημάτων, η οποία επίσης δε φαίνεται ;

Μία από τις λύσεις που δόθηκαν στο πρώτο πρόβλημα είναι η χρήση *δεικτών* (οι οποίοι καταγράφουν τη διεύθυνση της οντότητας που "δείχνουν" στο ιεραρχικό δένδρο) που επιτρέπουν σε μία οντότητα να προσπελάσει τις εξαρτώμενες από αυτήν οντότητες παρ' ότι αυτές μετέχουν ήδη σε συσχέτιση με άλλη οντότητα. Εντούτοις οι δείκτες, μολονότι βολικοί, αποτελούν μια όχι και τόσο ακριβή αναπαράσταση του ιδεατού μοντέλου. Η εκλογή της συσχέτισης που θα αναπαρασταθεί ως δείκτης (στο παράδειγμα μας είναι η Καθηγητής – Παράδοση), επηρεάζει και τη συμπεριφορά του μοντέλου.

Η λύση στο δεύτερο πρόβλημα είναι η επινόηση μιας νέας "τεχνητής" οντότητας η οποία θα έχει σχέση ένα προς πολλά (με τη νέα οντότητα να είναι από τη μεριά του πολλά), με κάθε μια από τις δύο οντότητες που μεταξύ τους είχαν σχέση πολλά προς πολλά. Το τροποποιημένο δένδρο απεικονίζεται στο σχήμα 1.7, στο οποίο η οντότητα "Εγγραφή" είναι η τεχνητή οντότητα με τη βοήθεια της οποίας απλοποιούμε τη σχέση πολλά προς πολλά των μαθητών με τα μαθήματα. Έχουμε δηλαδή τη σχέση ένα προς πολλά μεταξύ μαθητών και εγγραφών (ένας μαθητής εγγράφεται σε πολλά μαθήματα) και τη σχέση ένα προς πολλά μεταξύ μαθημάτων και εγγραφών (σε ένα μάθημα έχουν γίνει πολλές εγγραφές).

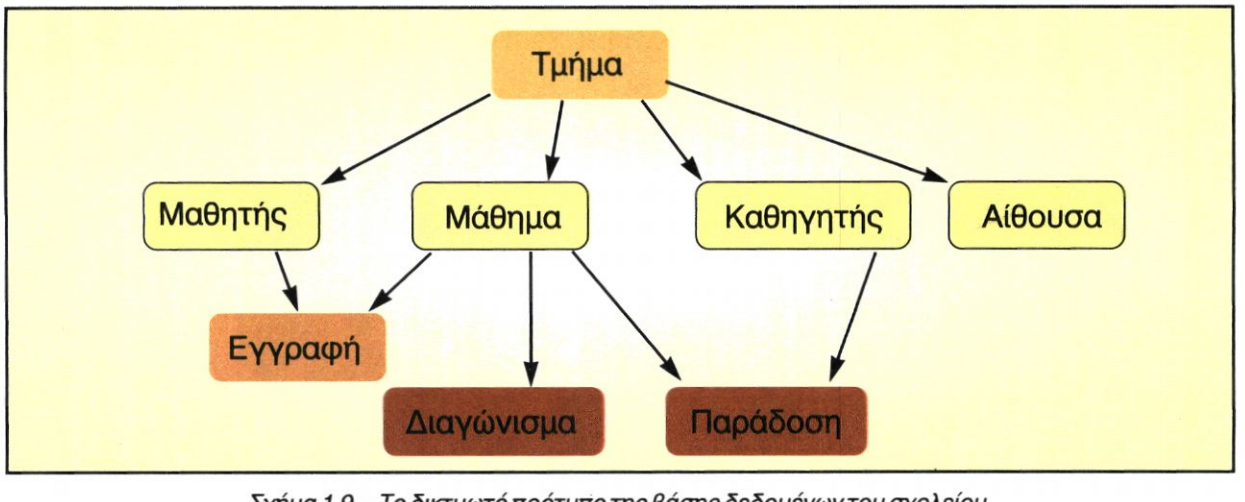
Το ιεραρχικό μοντέλο προσφέρει υψηλή ταχύτητα διαχείρισης των δεδομένων, αλλά είναι δύσκολο στη χρήση και στη συντήρηση και δε διαθέτει εργαλεία ταχείας ανάπτυξης εφαρμογών. Πρακτικά σήμερα έχει εγκαταλειφθεί και συναντάται μόνο σε μεγάλα συστήματα που διαχειρίζονται παλιές μεγάλες βάσεις δεδομένων που δεν έχουν ακόμα μεταφερθεί σε σύγχρονα ΣΔΒΔ (κυρίως σχεσιακά).

1.2.7. Δικτυωτά Συστήματα Διαχείρισης Βάσεων Δεδομένων

Το δικτυωτό μοντέλο (στο οποίο βασίζονται τα δικτυωτά ΣΔΒΔ), αναπτύχθηκε για να ξεπεραστούν τα προβλήματα τα οποία έπασχε το ιεραρχικό μοντέλο (π.χ., χρήση δεικτών κ.λ.π.). Αντί να βασίζεται σε μια οντότητα που θα αποτελέσει τη μοναδική ρίζα, το δικτυωτό μοντέλο βασίζεται σε επιμέρους συσχετίσεις.

Σχήμα 1.8. - Παράδειγμα ενός τύπου συνόλου.

Η δομική μονάδα του δικτυωτού μοντέλου είναι το *σύνολο*, το οποίο συντίθεται από δύο οντότητες που έχουν μεταξύ τους σχέση ένα προς πολλά. Η οντότητα από τη μεριά του ένα λέμε ότι είναι **ιδιοκτήτης** του συνόλου και η οντότητα από τη μεριά του πολλά λέμε ότι είναι *μέλος* του συνόλου. Στο **σχήμα 1.8** απεικονίζεται ένα παράδειγμα συνόλου παρμένο από τη βάση δεδομένων του σχολείου. Σαν όνομα διαλέγουμε κάτι που δηλώνει τη σχέση μεταξύ των οντοτήτων που μετέχουν σε αυτό (Μάθημα_διδασκόμενο_στο). Ολόκληρο το μοντέλο δεδομένων χτίζεται προσδιορίζοντας ένα σύνολο για κάθε συσχέτιση του ιδεατού μοντέλου μας και προσθέτοντας το στη συλλογή των συνόλων που έχουμε ήδη προσδιορίσει. Οι οντότητες μπορούν να συμμετέχουν σε περισσότερα από ένα σύνολα και έτσι τα σύνολα που απαρτίζουν το μοντέλο δεδομένων μπορούν να αλληλεπικαλύπτονται, σχηματίζοντας ένα "δίκτυο" (βλ. **σχήμα 1.9**).



Σχήμα 1.9. - Το δικτυωτό πρότυπο της βάσης δεδομένων του σχολείου.

Η κατασκευή του δικτυωτού μοντέλου γίνεται βάσει των ακόλουθων κανόνων:

1. Κανένα στιγμιότυπο μιας οντότητας δεν μπορεί να είναι ιδιοκτήτης περισσότερων από ένα συνόλων του ίδιου τύπου.
2. Κανένα στιγμιότυπο μιας οντότητας δεν μπορεί να είναι μέλος περισσότερων από ένα συνόλων του ίδιου τύπου.
3. Κανένα στιγμιότυπο μιας οντότητας δεν μπορεί να είναι ταυτόχρονα και ιδιοκτήτης και μέλος του ίδιου τύπου συνόλου.

Με βάση τα όσα έχουμε πει, το δικτυωτό μοντέλο της βάσης δεδομένων του σχολείου απεικονίζεται στο σχήμα 1.9. Σε αντίθεση με το σχήμα 1.6, όπου απεικονίζονται στιγμιότυπα οντοτήτων, εδώ απεικονίζουμε μόνο τους τύπους συνόλων που απαρτίζουν το μοντέλο δεδομένων. Η απεικόνιση των στιγμιότυπων στο δικτυωτό μοντέλο δε θα μας απασχολήσει σε αυτό το βιβλίο.

Όπως έχουμε πει και προηγουμένως, το σύνολο αναπαριστά μόνο σχέση ένα προς πολλά μεταξύ οντοτήτων. Άρα είναι φανερό ότι και στο δικτυωτό μοντέλο εξακολουθούμε να χρειαζόμαστε την επινόηση μιας νέας τεχνητής οντότητας (της "Εγγραφής", βλ. και ενότητα 1.2.6 προκειμένου να μπορέσουμε να αναπαραστήσουμε τη σχέση πολλά προς πολλά μεταξύ μαθητών – μαθημάτων. Αυτή η οντότητα είναι μέλος δύο συνόλων, ιδιοκτήτες των οποίων είναι οι οντότητες που μεταξύ τους έχουν σχέση πολλά προς πολλά ("Μαθητής" και "Μάθημα" στο παράδειγμά μας).

Το δικτυωτό μοντέλο ξεκίνησε στις αρχές της δεκαετίας του '70 από την πρόταση που έκανε το Διεθνές Συνέδριο Γλωσσών Συστημάτων Δεδομένων (C.O.DA.SY.L. - Conference On Data System Languages) για ένα μοντέλο στα συστήματα βάσεων δεδομένων. Στο μοντέλο αυτό έγινε για πρώτη φορά η διάκριση της γλώσσας διαχείρισης της βάσης δεδομένων σε δύο ενότητες τη Γλώσσα Ορισμού Δεδομένων και τη Γλώσσα Διαχείρισης Δεδομένων. Η διάκριση αυτή διατηρήθηκε και αργότερα στο σχεσιακό μοντέλο.

Το δικτυωτό μοντέλο είναι πιο εύκολο στη χρήση και στη συντήρηση από το ιεραρχικό μοντέλο, ενώ προσφέρει την ίδια υψηλή ταχύτητα διαχείρισης των δεδομένων. Δεν πρόλαβε όμως να εδραιωθεί, γιατί σύντομα μετά την εμφάνισή του ακολούθησε η εμφάνιση του σχεσιακού μοντέλου που δέσποσε (και δεσπόζει) στην αγορά.

Παραδείγματα δικτυωτών ΣΔΒΔ αποτελούν τα IDS II της Honeywell/Bull, DMS II της Burroughs, DMS 1100 της Unisys και το IMAGE της Hewlett-Packard.

Ανακεφαλαίωση / περίληψη

Το λογισμικό είναι τα προγράμματα που κατευθύνουν τη λειτουργία του υπολογιστή και επιτελούν τη διαχείριση και επεξεργασία των δεδομένων, μαζί με όλη τη βοηθητική τεκμηρίωση που το συνοδεύει (εγχειρίδια χρήσης, πηγαίος κώδικας κ.λ.π.). Μία δυνατή κατάταξη του λογισμικού είναι η εξής: λογισμικό συστήματος, το επιχειρησιακό λογισμικό, επιστημονικό λογισμικό, βιομηχανικό λογισμικό, λογισμικό στις μεταφορές και το ψυχαγωγικό λογισμικό.

Τα δεδομένα που χρειάζονταν αρχικά οι διάφορες εφαρμογές λογισμικού τα διαχειρίζονταν αρχικά οι ίδιες και τα αποθήκευαν σε αρχεία δικά τους. Με την εμφάνιση όμως των συστημάτων διαχείρισης βάσεων δεδομένων (ΣΔΒΔ), τα δεδομένα συγκεντρώθηκαν σε βάσεις δεδομένων και η διαχείρισή τους γινόταν κεντρικά από τα ΣΔΒΔ, διότι αυτά πλεονεκτούσαν σε σχέση τις παλιές εφαρμογές διαχείρισης αρχείων στην επεκτασιμότητα, στη συντήρηση και στην ανεξαρτησία δεδομένων.

Το ΣΔΒΔ ασχολείται με τη λογική οργάνωση των δεδομένων, δηλαδή το τι αντιπροσωπεύουν, πώς ονομάζονται, τι τιμή έχουν, ποια είναι η μεταξύ τους σχέση. Το ΣΔΒΔ σχηματίζει ένα μοντέλο του πραγματικού συστήματος (π.χ., σχολείο, επιχείρηση κ.λ.π.) το οποίο αναπαρίσταται στο ιδεατό σχήμα της βάσης δεδομένων. Στο μοντέλο αυτό, κάθε συστατικό του υπό εξέταση πραγματικού συστήματος που έχει αυτόνομη ύπαρξη και για το οποίο μας ενδιαφέρει να συλλέξουμε πληροφορίες το ονομάζουμε οντότητα. Το είδος των δομών δεδομένων που χρησιμοποιεί το κάθε ΣΔΒΔ ως δομικό λίθο για την κατασκευή του ιδεατού σχήματος, αναφέρεται και σαν μοντέλο δεδομένων και ανάλογα με αυτό τα ΣΔΒΔ χωρίζονται σε ιεραρχικά, δικτυωτά, σχεσιακά.

Τα σχεσιακά ΣΔΒΔ είναι τα πλέον διαδεδομένα στην αγορά παγκόσμια και βασίζονται στην απλούστερη των δομών δεδομένων, τον πίνακα. Κάθε στήλη του πίνακα αντιστοιχεί σε μια ιδιότητα (μιας οντότητας ή μιας συσχέτισης) και το σύνολο των τιμών των στηλών σε κάθε γραμμή του πίνακα αντιστοιχεί σε ένα στιγμιότυπο μιας οντότητας ή μιας συσχέτισης. Το διεθνές πρότυπο για την επικοινωνία του χρήστη και του προγραμματιστή με όλα τα σχεσιακά ΣΔΒΔ είναι η γλώσσα SQL.

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Κατατάξτε το λογισμικό του σχολικού εργαστηρίου σε κάποια (ή κάποιες) από τις κατηγορίες λογισμικού που γνωρίζετε.
2. Ποια η διαφορά προγράμματος – λογισμικού;
3. Ποια τα πλεονεκτήματα - μειονεκτήματα των συστημάτων διαχείρισης βάσεων δεδομένων σε σχέση με τα συστήματα διαχείρισης αρχείων;

4. Περιγράψτε τα βήματα σχηματισμού ενός προτύπου βάσης δεδομένων
5. Γιατί επικράτησε το σχεσιακό μοντέλο ΣΔΒΔ απέναντι στο ιεραρχικό και το δικτυωτό;
6. Από τις παρακάτω λέξεις ποιες εκφράζουν οντότητα σε ένα σύστημα διαχείρισης σχολείου;
   - ❑ Μαθητής
   - ❑ Ονοματεπώνυμο μαθητή
   - ❑ Έτος σπουδών μαθητή
   - ❑ Καθηγητής
   - ❑ Ειδικότητα καθηγητή
   - ❑ Μάθημα διδασκόμενο από τον καθηγητή
7. Με βάση τον πίνακα 1.2 διατυπώστε μια απλή σχέση σε μορφή πίνακα για την οντότητα "δίσκος" σε ένα σύστημα διαχείρισης δισκοθήκης.

Γλωσσάριο - Ευρετήριο όρων

αλγόριθμος *(algorithm)* - Μια προσδιορισμένη με ακρίβεια ακολουθία εντολών για την επίλυση ενός προβλήματος.
βάση δεδομένων - Μια συλλογή δεδομένων οργανωμένη με σαφήνεια [25].
δομημένη γλώσσα ερωτήσεων *(Structured Query Language - SQL)* - Πρότυπη γλώσσα για την επικοινωνία του χρήστη και του προγραμματιστή με όλα τα σχεσιακά ΣΔΒΔ [31].
λογισμικό *(software)* - Οι αλγόριθμοι, οι δομές δεδομένων και τα κείμενα που περιγράφουν τη χρήση και τις λειτουργίες των προγραμμάτων [19].
οντότητα *(entity)* - Κάθε συστατικό ενός υπό εξέταση συστήματος, για το οποίο μας ενδιαφέρει να συλλέξουμε πληροφορίες Σε μία βάση δεδομένων αναφέρεται στο είδος της πληροφορίας που αποθηκεύεται, π.χ ένα πρόσωπο, μία έννοια, ένα γεγονός. Η οντότητα όταν χρησιμοποιείται κατά το σχεδιασμό μίας βάσης δεδομένων, αποτελεί τον πρόγονο ενός πίνακα [29].
μοντέλο δεδομένων *(data model)* - Το είδος των δομών δεδομένων που χρησιμοποιεί το κάθε ΣΔΒΔ. Με βάση αυτό, τα ΣΔΒΔ χωρίζονται σε ιεραρχικά, δικτυωτά και σχεσιακά [30].

Βιβλιογραφία

Bowers, D.S., "From Data to Database", Second Edition, Chapman & Hall 1993.
Pressman, R.S., "Software Engineering: A Beginner's Guide", McGraw-Hill, 1988.
Vossen G., "Data Models, Database Languages and Database Management Systems", Addison Wesley 1991.
Γιαλούρη Κ. και άλλων, "Εφαρμογές Πληροφορικής - Υπολογιστών Α', Β', Γ' Ενιαίου Λυκείου", ΟΕΔΒ 1998.
Δέρβου Δ., "Μαθήματα Βάσεων Δεδομένων", Εκδόσεις Α. Τζιόλα 1995.

ΚΕΦΑΛΑΙΟ 2

ΕΠΙΚΟΙΝΩΝΙΑ – ΣΥΝΕΡΓΑΣΙΑ ΕΦΑΡΜΟΓΩΝ ΛΟΓΙΣΜΙΚΟΥ

ΠΕΡΙΕΧΟΜΕΝΑ

2.1. Εκμετάλλευση δεδομένων από άλλες εφαρμογές

2.2. Μετατροπές αρχείων

2.3. Τεχνικές συνεργασίας εφαρμογών

ΕΙΣΑΓΩΓΗ

Η ανάγκη επικοινωνίας διαφορετικών εφαρμογών που χρειάζονται κατά καιρούς να διαχειριστούν κοινά δεδομένα, η που επεξεργάζονται διαδοχικά τα ίδια δεδομένα, προέκυψε πολύ νωρίς, από την εποχή των μεγάλων κεντρικών υπολογιστών.

- Πως είναι δυνατό διαφορετικές εφαρμογές να χρησιμοποιούν την ίδια βάση δεδομένων, ιδιαίτερα όταν αυτή είναι εγκαταστημένη σε άλλον υπολογιστή από τις εφαρμογές;

Με την εξάπλωση των δικτύων υπολογιστών και των γραφικών διεπαφών επικοινωνίας του χρήστη με τον υπολογιστή, η ανταλλαγή δεδομένων μεταξύ εφαρμογών έγινε απαραίτητη (π.χ., ενσωμάτωση εικόνων σε έγγραφα επεξεργαστών κειμένου, σύνδεση φύλλων εργασίας με βάσεις δεδομένων κ.λ.π.).

- Πώς γίνεται αυτό ιδιαίτερα όταν οι εφαρμογές είναι κατασκευασμένες από άλλους κατασκευαστές;

Αυτά καθορίζονται από διάφορα πρωτόκολλα που επιτρέπουν όχι μόνο την ανταλλαγή δεδομένων αλλά και την εξ' αποστάσεως κλήση συναρτήσεων της μιας εφαρμογής από την άλλη. Σε αυτό το κεφάλαιο θα εξετάσουμε τις βασικές τεχνικές και πρωτόκολλα μέσω των οποίων γίνεται η συνεργασία των εφαρμογών στους σύγχρονους υπολογιστές.

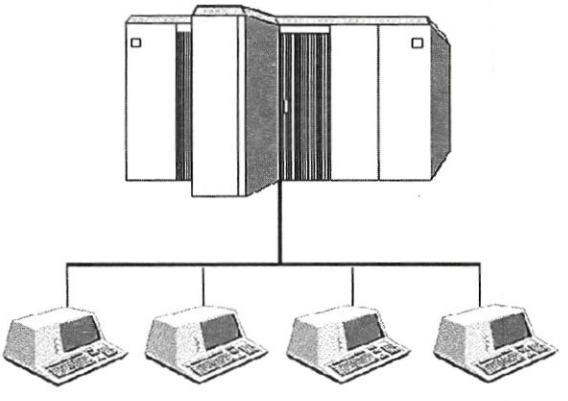
ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να αναγνωρίζει ο μαθητής τις βασικές τεχνικές και τα πρωτόκολλα, μέσω των οποίων γίνεται η συνεργασία των εφαρμογών στους υπολογιστές.
2. Να διακρίνει με ποιους τρόπους μετατρέπουμε ήδη μορφοποιημένα δεδομένα από ένα σύστημα ή μία εφαρμογή σε άλλο σύστημα ή εφαρμογή.
3. Να αξιοποιεί τις γνώσεις του στην πράξη συσχετίζοντας δεδομένα μεταξύ διαφορετικών εφαρμογών, επιλέγοντας το προσφορότερο τρόπο.

2.1. ΕΚΜΕΤΑΛΛΕΥΣΗ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΑΛΛΕΣ ΕΦΑΡΜΟΓΕΣ

2.1.1. Από τους κεντρικούς υπολογιστές στα τοπικά δίκτυα

Μέχρι την εξάπλωση των προσωπικών υπολογιστών και των τοπικών δικτύων στα μέσα της δεκαετίας του 1980, στις επιχειρήσεις και στους οργανισμούς κυριαρχούσαν οι κεντρικοί υπολογιστές, είτε ως mainframes, είτε ως minicomputers. Όπως φαίνεται και στο απλουστευμένο **σχήμα 2.1,** καρδιά του συστήματος ήταν ο κεντρικός υπολογιστής (που ακόμα και όταν ήταν "minicomputer" είχε το μέγεθος μιας ντουλάπας) και οι χρήστες επικοινωνούσαν με αυτόν μέσω "χαζών" τερματικών χαρακτήρων.



Σχήμα 2.1. - Κεντρικός υπολογιστής με τερματικά.

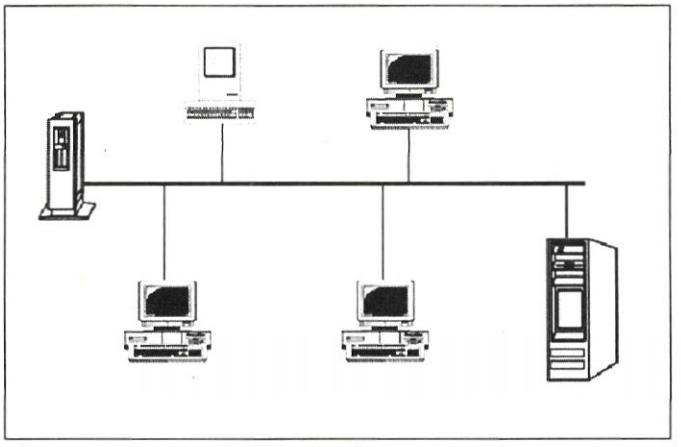
Το επίθετο "χαζά" δεν αναφέρεται φυσικά στην ευφυία των μηχανών, αλλά στο γεγονός ότι τα τερματικά έχουν σαν αποκλειστικό καθήκον την επικοινωνία με το χρήστη και δε συμμετέχουν στην επεξεργασία των δεδομένων. Όπως είπαμε τα τερματικά αυτά πρόβαλλαν μόνο αλφαριθμητικούς χαρακτήρες και κάποιους ειδικούς χαρακτήρες για πλαίσια (π.χ., οριζόντιες και κάθετες γραμμές) και δεν είχαν τη δυνατότητα προβολής γραφικών. Ο τρόπος χειρισμού τους εξαρτιόταν από την εταιρία κατασκευής και το μοντέλο και ορισμένες φορές ήταν ειδικά προσαρμοσμένα στην εφαρμογή που εξυπηρετούσαν. Κάτω απ' αυτές τις συνθήκες η δυνατότητα μοιράσματος δεδομένων μεταξύ διαφορετικών εφαρμογών ήταν εξαιρετικά περιορισμένη, δύσκολα υλοποιήσιμη και συνήθως απαιτούσε και αλλαγές στον κώδικα των εφαρμογών που θα μοιράζονταν τα δεδομένα.

Με τα παραπάνω δε θέλουμε να στείλουμε τα συστήματα κεντρικού υπολογιστή στην πυρά, απλά επισημάναμε αδυναμίες αυτής της αρχιτεκτονικής και ιδιαίτερα του σταδίου ανάπτυξης που βρισκόταν τότε. Τα συστήματα αυτά όμως ήταν εξαιρετικά αξιόπιστα (ήταν σχεδιασμένα να εργάζονται 365 ημέρες το χρόνο 24 ώρες την ημέρα), ασφαλή και αποτελεσματικά στην επεξεργασία μεγάλου όγκου δεδομένων.

Η διαρκής αύξηση της υπολογιστικής ισχύος των σταθμών εργασίας και των προσωπικών υπολογιστών με παράλληλη μείωση του κόστους και η εξάπλωση των τοπικών δικτύων, άλλαξαν τα υπολογιστικά συστήματα των επιχειρήσεων σε μια μορφή που απεικονίζεται στο **σχήμα 2.2**.

Τους μεγάλους κεντρικούς υπολογιστές αντικατέστησαν σε πολλές περιπτώσεις τοπικά δίκτυα προσωπικών υπολογιστών και ισχυρότερων σταθμών εργασίας.

Υπολογιστές από διαφορετικούς κατασκευαστές, με διαφορετική αρχιτεκτονική, που λειτουργούν με διαφορετικά λειτουργικά συστήματα επικοινωνούν μέσω τοπικού δικτύου.



Σχήμα 2.2. - Δίκτυο από υπολογιστές διαφορετικής αρχιτεκτονικής.

Οι υπολογιστές στην κάθε θέση εργασίας δεν παίζουν πια το ρόλο των "χαζών" τερματικών, αλλά έχουν επαρκή ισχύ για να εκτελούν ένα μέρος (ή και το σύνολο καμία φορά) των εφαρμογών. Οι πιο ισχυροί από αυτούς φιλοξενούν τα συστήματα διαχείρισης βάσεων δεδομένων (στη συντριπτική πλειοψηφία τους σχεσιακά) και είναι οι **εξυπηρετητές** *(servers)*. Αυτοί επεξεργάζονται τα δεδομένα ανταποκρινόμενοι σε αιτήσεις των εφαρμογών που εκτελούνται στους υπολογιστές στις θέσεις εργασίας, οι οποίοι ονομάζονται και **πελάτες** *(clients)*. Οι εξυπηρετητές μέσω του δικτύου στέλνουν στους πελάτες το αποτέλεσμα της επεξεργασίας (π.χ., προσθήκη, διαγραφή, αναζήτηση κ.λ.π.) των δεδομένων. Ανάλογα δε με το είδος των εφαρμογών που εκτελούν οι πελάτες, το αποτέλεσμα αυτό το προβάλλουν κατ' ευθείαν στην οθόνη, ή το επεξεργάζονται τοπικά (π.χ., ζωγραφίζουν διαγράμματα κ.λ.π.). Τα συστήματα αυτά ονομάζονται συστήματα **πελάτη – εξυπηρετητή** *(client – server)*. Η πλειοψηφία των μεσαίων – μεγάλων σχεσιακών συστημάτων διαχείρισης βάσεων δεδομένων είναι εγκατεστημένα και λειτουργούν σε τέτοια συστήματα, όπου το ρόλο των εξυπηρετητών αναλαμβάνουν από ισχυροί προσωπικοί υπολογιστές, έως minicomputers με 64 παράλληλους επεξεργαστές.

Μια οικονομικότερη εκδοχή απέναντι στα συστήματα πελάτη – εξυπηρετητή*πελάτη – εξυπηρετητή*, είναι τα συστήματα που βασίζονται σε **εξυπηρετητές (ή διακομιστές) αρχείων**. Και σε αυτά υπάρχουν ετερογενή μηχανήματα και λειτουργικά συστήματα (**σχήμα 2.2**), αλλά οι εξυπηρετητές εδώ προσφέρουν μόνο το χώρο στο δίσκο τους στους πελάτες και δεν φιλοξενούν συστήματα διαχείρισης δεδομένων. Αυτό έχει σα συνέπεια ο κάθε πελάτης να έχει το δικό του σύστημα διαχείρισης της βάσεως δεδομένων και απλά να διαβάζει από το δίσκο του εξυπηρετητή σα να πρόκειται για το δικό του δίσκο. Τα συστήματα αυτά επιβαρύνουν το τοπικό δίκτυο γιατί η επεξεργασία γίνεται στον πελάτη, με αποτέλεσμα να μεταφέρει το σύνολο των δεδομένων από τον εξυπηρετητή αρχείων για να το επεξεργαστεί, σε αντίθεση με τα συστήματα πελά-

τη – εξυπηρετητή στα οποία ο εξυπηρετητής στέλνει μόνο το αποτέλεσμα της επεξεργασίας στον πελάτη.

Γενικά τα συστήματα με εξυπηρετητή αρχείων έχουν χαμηλότερες επιδόσεις και επεκτασιμότητα από τα συστήματα πελάτη - εξυπηρετητή, αλλά και χαμηλότερο κόστος λειτουργίας και διαχείρισης. Μεγάλη δημοτικότητα γνώρισαν σε δίκτυα προσωπικών υπολογιστών στο τέλος της δεκαετίας του '80 και στο πρώτο μισό της δεκαετίας του '90, όπου εκτελούσαν εφαρμογές κατασκευασμένες κυρίως με Clipper, ή με κάποια από τα δημοφιλή προσωπικά ΣΔΒΔ της εποχής (Dbase IV, Paradox, Access κ.λ.π.).

2.1.2. Ανοιχτά συστήματα

Τα συστήματα πελάτη – εξυπηρετητή *πελάτη – εξυπηρετητή* που περιγράψαμε στην προηγούμενη παράγραφο δε θα μπορούσαν να εξαπλωθούν παρά μόνο αν λυνόταν ένα πρόβλημα που μέχρι τότε δεν είχε απασχολήσει ιδιαίτερα τις εταιρίες κατασκευής υπολογιστών.

Για τη σωστή αξιοποίηση της αρχιτεκτονικής πελάτη – εξυπηρετητή, έπρεπε να σταθεί δυνατό να συνεργαστούν εξυπηρετητές και πελάτες διαφορετικών κατασκευαστών, σχεδιασμένοι για διαφορετικά λειτουργικά συστήματα. Αυτό οδήγησε στην εμφάνιση των ανοιχτών συστημάτων.

Καθώς όπως είδαμε συνδύαζαν υπολογιστές διαφορετικής αρχιτεκτονικής, οι οποίοι λειτουργούσαν κάτω από διαφορετικά λειτουργικά συστήματα, έπρεπε να βρεθεί ένας πρότυπος τρόπος επικοινωνίας αυτών των εφαρμογών. Ακόμα θα έπρεπε για παράδειγμα, να μπορεί να εξασφαλιστεί ότι αν κάποιος που χρησιμοποιούσε ένα σχεσιακό ΣΔΒΔ μιας εταιρίας Χ στον εξυπηρετητή του με λειτουργικό σύστημα Α, αποφάσιζε να αναβαθμίσει τον εξυπηρετητή σε έναν άλλο ισχυρότερο που χρησιμοποιούσε λειτουργικό σύστημα Β και σχεσιακό ΣΔΒΔ της ίδιας εταιρίας Χ (ή ακόμη και άλλης εταιρίας Υ), θα μπορούσε να κάνει αυτήν την αναβάθμιση χωρίς να αλλάξει τις εφαρμογές των πελατών, ή το τοπικό του δίκτυο. Επιπρόσθετα, ο κώδικας διαχείρισης των εφαρμογών που ήταν ενσωματωμένος στο παλιό ΣΔΒΔ θα έπρεπε να μπορούσε να μεταφερθεί στο νέο, χωρίς να χρειαστεί να ξαναγραφεί, ή έστω χωρίς να χρειαστεί να ξαναγραφεί εξ ολοκλήρου.

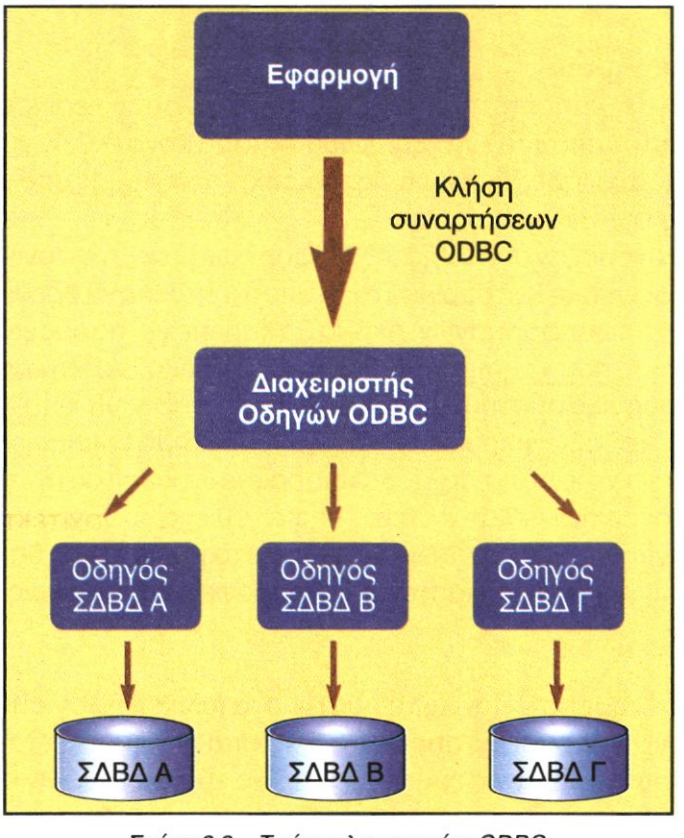
Ορισμός

Ως **ανοιχτά** χαρακτηρίζονται τα συστήματα (υλικού ή λογισμικού) που αναπτύσσονται σύμφωνα με ευρέως αποδεκτά πρότυπα ή τεχνικές, έτσι ώστε να μπορούν να ανταλλάσσουν τμήματά τους με αντίστοιχα τμήματα άλλων κατασκευαστών χωρίς να διαταράσσεται η λειτουργία τους.

Το επίθετο "πρότυπος" με το οποίο χαρακτηρίσαμε τον τρόπο επικοινωνίας των εφαρμογών, αφορά στην ανεξαρτησία που θα πρέπει να έχει από την εμπορική πολιτική μιας συγκεκριμένης εταιρίας. Η στενή εξάρτηση που είχαν τα συστήματα κεντρικού υπολογιστή από την κατασκευάστρια εταιρία ήταν ένας από τους λόγους για τους οποίους το κόστος τους ήταν τόσο υψηλό.

Ως ανταπόκριση στην ανάγκη αυτή θεσπίστηκαν διάφορες προδιαγραφές από διεθνείς οργανισμούς, ή από ομάδες εταιριών. Ένα παράδειγμα είναι η **Ανοικτή Διασυνδεσιμότητα Βάσεων Δεδομένων –** *Open Database Connectivity (ODBC).* Αυτή είναι στην ουσία ένα σύνολο συναρτήσεων και δομών δεδομένων που δίνει τη δυνατότητα σε μια εφαρμογή να ανακτήσει δε-

δομένα που βρίσκονται αποθηκευμένα σε βάσεις δεδομένων διαφόρων κατασκευαστών, ανεξάρτητα από τα συγκεκριμένα χαρακτηριστικά της κάθε μιας από αυτές, αν είναι σχεσιακές ή όχι κ.λ.π. Ο τρόπος λειτουργίας της φαίνεται στο **σχήμα 2.3.** Η εφαρμογή καλεί μέσω προκαθορισμένων συναρτήσεων από τη βιβλιοθήκη της ODBC το διαχειριστή οδηγών ODBC. Αυτός αναγνωρίζει το ΣΔΒΔ στον οποίο θέλει να απευθυνθεί η εφαρμογή και αποστέλλει κατάλληλες εντολές SQL στο πρόγραμμα – οδηγό του ΣΔΒΔ, το οποίο με τη σειρά του τις μεταφράζει στη συγκεκριμένη εκδοχή της γλώσσας που χρησιμοποιεί το ΣΔΒΔ.



Σχήμα 2.3. - Τρόπος λειτουργίας ODBC.

Η ODBC βασίστηκε στις προδιαγραφές Διεπαφής Επιπέδου Κλήσης – Call Level Interface (CLI) που διατυπώθηκαν το 1992 από την ομάδα SQL Access Group. Αυτή σχηματίστηκε το 1992 από μερικές από τις μεγαλύτερες αμερικανικές εταιρίες πληροφορικής, όπως η AT&T, η IBM, η Informix, η Microsoft, η Sybase και άλλες.

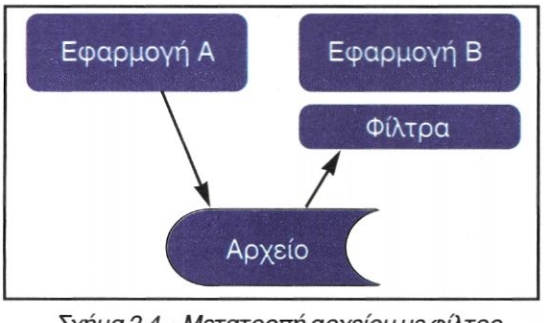
Η ODBC έχει κατηγορηθεί για χαμηλές επιδόσεις, αν και αυτό εξαρτάται από τη συγκεκριμένη υλοποίηση και το λειτουργικό σύστημα. Όπως και να έχει, αποτελεί ένα ευρύτατα διαδεδομένο πρότυπο, ιδιαίτερα σε προσωπικούς υπολογιστές.

Ένα άλλο παράδειγμα είναι η Μεταφέρσιμη Διεπαφή Λειτουργικού Συστήματος - POSIX (Portable Operating System Interface), ένα σύνολο προδιαγραφών που θεσπίστηκαν από το Ινστιτούτο Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών (IEEE), για να εξασφαλίσουν την μεταφερσιμότητα των εφαρμογών μεταξύ διαφόρων παραλλαγών του λειτουργικού συστήματος UNIX, αλλά τμήματα τους υιοθετήθηκαν επίσης και από κατασκευαστές άλλων λειτουργικών συστημάτων.

2.2. ΜΕΤΑΤΡΟΠΕΣ ΑΡΧΕΙΩΝ

2.2.1. Φίλτρα

Ο παραδοσιακός τρόπος για ανταλλαγή δεδομένων μεταξύ εφαρμογών είναι η ανταλλαγή αρχείων. Αυτή μπορεί να συμβαίνει μεταξύ εφαρμογών εγκαταστημένων στον ίδιο υπολογιστή, ή μπορεί να συμβαίνει μεταξύ εφαρμογών εγκατεστημένων σε διαφορετικούς υπολογιστές. Στη δεύτερη περίπτωση μεταφέρουμε το αρχείο μέσω ενός φορητού μέσου (π.χ., δισκέτες), ή το στέλνουμε μέσω ενός καναλιού επικοινωνίας (π.χ., τηλεφωνική σύνδεση).



Σχήμα 2.4. - Μετατροπή αρχείου με φίλτρο.

Τα αρχεία αυτά που ανταλλάσσονται μεταξύ των εφαρμογών πρέπει να είναι σε μορφή αντιληπτή από αυτές, πράγμα όχι πάντα εύκολο, ιδιαίτερα αν οι εφαρμογές είναι εγκατεστημένες σε υπολογιστές διαφορετικής αρχιτεκτονικής. Συνήθως το αρχείο που παράγεται από μια εφαρμογή πρέπει να επεξεργαστεί από ένα ειδικό λογισμικό, το λεγόμενο **φίλτρο** για να αποκτήσει μια δομή συμβατή με την άλλη εφαρμογή (βλ. **σχήμα 2.4**).

Ανάμεσα στις λειτουργίες που εκτελεί ένα φίλτρο, είναι η αλλαγή της λογικής οργάνωσης των δεδομένων μέσα στο αρχείο, ή αλλαγή της κωδικοποίησης των χαρακτήρων και η αλλαγή του τρόπου φυσικής αποθήκευσης των αριθμητικών δεδομένων στο αρχείο.

2.2.2. Λειτουργίες μετατροπής

Τα δεδομένα στο αρχείο είναι αποθηκευμένα με τη μορφή εγγραφών (*records*). Χονδρικά μπορούμε να αντιστοιχίσουμε αυτές τις εγγραφές με τις οντότητες του μοντέλου δεδομένων. Ο τρόπος με τον οποίο είναι αποθηκευμένες οι εγγραφές (π.χ., σειριακά, με ευρετήρια κ.λ.π.) καθορίζει και την λογική οργάνωση του αρχείου (περισσότερα για τον τρόπο οργάνωσης των δεδομένων σε αρχεία θα δούμε στο κεφάλαιο 6). Το φίλτρο μπορεί να χρειαστεί να αλλάξει τη λογική οργάνωση του αρχείου, ή να κάνει επιμέρους αλλαγές όπως η αλλαγή του μεγέθους των πεδίων μιας εγγραφής, η αλλαγή του μήκους της εγγραφής, αλλαγή του διαχωριστικού των πεδίων κ.λ.π.

Ένα άλλο σημαντικό πρόβλημα (ιδιαίτερα σε εφαρμογές σε γλώσσες εκτός της αγγλικής) είναι η ανάγκη για μετατροπή στην κωδικοποίηση των χαρακτήρων. Σε κάθε υπολογιστικό σύστημα οι χαρακτήρες του αλφαβήτου τα σημεία στίξεως καθώς και άλλοι ειδικοί χαρακτήρες (π.χ., σύμβολα νομισμάτων, αλλαγή γραμμής κ.λ.π.) αντιστοιχίζονται (κωδικοποιούνται) μονοσήμαντα με ακέραιους αριθμούς, τους οποίους μεταφράζει ο υπολογιστής σε αναγνωρίσιμους χαρακτήρες στην οθόνη ή στο χαρτί του εκτυπωτή.

Το πλέον διαδεδομένο πρότυπο κωδικοποίησης χαρακτήρων είναι το πρότυπο **ASCII** (Αμερικάνικος Πρότυπος Κώδικας Ανταλλαγής Πληροφοριών). Όπως βλέπουμε και στον **πίνακα 2.1**, αξιοποιεί τα 7 χαμηλότερα bits (0-127) από έναν ακέραιο των 8 bits (0-255). Οι υπόλοιπες 128 θέσεις είναι διαθέσιμες για κωδικοποίηση χαρακτήρων από άλλες γλώσσες (π.χ., ελληνικά) καθώς και για ειδικούς γραφικούς χαρακτήρες (κάθετες – οριζόντιες γραμμές κ.λ.π.).

Ακόμα και για την ίδια γλώσσα, ο τρόπος κωδικοποίησής της στις επεκταμένες θέσεις του Πίνακα ASCII (128-255) έχει πολλές φορές εναλλακτικές μορφές, οι οποίες ορίζονται και από de facto ή de jure πρότυπα. Για παράδειγμα υπάρχουν για τα ελληνικά το πρότυπο ΕΛΟΤ 928, η κωδικοποίηση IBM 737 που ήταν συνηθισμένη σε προσωπικούς υπολογιστές με DOS κ.λ.π.

Εκτός από το πρότυπο ASCII που έχει επικρατήσει στους περισσότερους υπολογιστές, υπάρχει και η κωδικοποίηση EBCDIC (Εκτεταμένος Δυαδικά Κωδικοποιημένος Δεκαδικός Κώδικας Ανταλλαγών). Αυτή ορίστηκε από την IBM και χρησιμοποιείται στα mainframes της ίδιας εταιρίας. Υπάρχουν πολλές παραλλαγές της, για να υποστηρίξουν και άλλες γλώσσες πλην της αγγλικής.

Ανάλογα με την αρχιτεκτονική του επεξεργαστή (π.χ., Intel ή Motorola), ο τρόπος αποθήκευσης δυαδικών πληροφοριών (π.χ., ακέραιους αριθμούς μήκους 16 bits) σε αρχεία διαφέρει. Η διαφορά υπάρχει στη σειρά αποθήκευσης της ακολουθίας των bytes. Για παράδειγμα ο δεκαεξαδικός αριθμός 4F52 μπορεί να αποθηκευτεί ως 4F52 ή ως 524F, δηλαδή ο πρώτος τρόπος αποθηκεύει πρώτα το πλέον σημαντικό byte (4F δεκαεξαδικό) και ακολούθως το λιγότερο σημαντικό (52 δεκαεξαδικό), ενώ ο δεύτερος τρόπος κάνει το αντίστροφο.

Τα mainframes IBM/370, οι περισσότεροι επεξεργαστές RISC και οι υπολογιστές με επεξεργαστές Motorola (π.χ., Apple) χρησιμοποιούν τον πρώτο τρόπο (big-endian), ενώ οι υπολογιστές με επεξεργαστή Intel & Compaq Alpha τον δεύτερο (little endian). Αυτό έχει σαν αποτέλεσμα δυαδικό αρχείο παραγόμενο από υπολογιστή της πρώτης κατηγορίας να χρειάζεται κατάλληλη μετατροπή όταν πρόκειται να διαβαστεί από υπολογιστή της δεύτερης.

---

10 δικός	16 δικός	Χαρ.	10 δικός	16 δικός	Χαρ.	10 δικός	16 δικός	Χαρ.
0	00	NUL	43	2b	+	86	56	V
1	01	SOH	44	2c	,	87	57	W
2	02	STX	45	2d	-	88	58	X
3	03	ETX	46	2e	.	89	59	Y
4	04	EOT	47	2f	/	90	5a	Z
5	05	ENQ	48	30	0	91	5b	[
6	06	ACK	49	31	1	92	5c	\
7	07	BEL	50	32	2	93	5d	]
8	08	BS	51	33	3	94	5e	^
9	09	HT	52	34	4	95	5f	_
10	0a	LF	53	35	5	96	60	`
11	0b	VT	54	36	6	97	61	a
12	0c	FF	55	37	7	98	62	b
13	0d	CR	56	38	8	99	63	c
14	0e	SO	57	39	9	100	64	d
15	0f	SI	58	3a	:	101	65	e
16	10	DLE	59	3b	;	102	66	f
17	11	DC1	60	3c	<	103	67	g
18	12	DC2	61	3d	=	104	68	h
19	13	DC3	62	3e	>	105	69	i
20	14	DC4	63	3f	?	106	6a	j
21	15	NAK	64	40	@	107	6b	k
22	16	SYN	65	41	A	108	6c	l
23	17	ETB	66	42	B	109	6d	m
24	18	CAN	67	43	C	110	6e	n
25	19	EM	68	44	D	111	6f	o
26	1a	SUB	69	45	E	112	70	p
27	1b	ESC	70	46	F	113	71	q
28	1c	FS	71	47	G	114	72	r
29	1d	GS	72	48	H	115	73	s
30	1e	RS	73	49	I	116	74	t
31	1f	US	74	4a	J	117	75	u
32	20	SP	75	4b	K	118	76	v
33	21	!	76	4c	L	119	77	w
34	22	"	77	4d	M	120	78	x
35	23	#	78	4e	N	121	79	y
36	24	$	79	4f	O	122	7a	z
37	25	%	80	50	P	123	7b	{
38	26	&	81	51	Q	124	7c	|
39	27	'	82	52	R	125	7d	}
40	28	(	83	53	S	126	7e	~
41	29	)	84	54	T	127	7f	DEL
42	2a	*	85	55	U			

Πίνακας 2.1. - Κωδικοποίηση ASCII.

2.1.5. Σύγκριση της ανταλλαγής αρχείων και της αρχιτεκτονικής πελάτη – εξυπηρετητή

Ο **πίνακας 2.2** παρουσιάζει μια συγκριτική παράθεση των πλεονεκτημάτων και μειονεκτημάτων των δυο τεχνικών.

	Ανταλλαγή Αρχείων	Συστήματα Πελάτη-Εξυπηρετητή
Κόστος υλικού	Μηδέν	Χαμηλό στον πελάτη, μέτριο στον εξυπηρετητή
Τηλεπικοινωνιακό κόστος	Μηδέν	Τοπικά δίκτυα (χαμηλό) ή δίκτυα ευρείας ζώνης (υψηλό)
Κόστος λογισμικού	Μηδέν	υψηλό
Κόστος ανάπτυξης	Υψηλό – Κατασκευή φίλτρων τροποποίηση εφαρμογών	Χαμηλό (εμπεριέχεται στο κόστος λογισμικού)
Ευκολία χρήσης	Δύσκολα υλοποιήσιμη, εύκολο στη χρήση	Μεγάλη
Συμφωνία δεδομένων	Προβληματική	Πολύ μεγάλη (σε πραγματικό χρόνο)
Επικαιρότητα	Φτωχή	Πολύ καλή
Πλεονάζοντα δεδομένα	Πολλά	Ελάχιστα
Κόστος διαχείρισης	Μεταβαλλόμενο	Σημαντικό

Πίνακας 2.2. - Σύγκριση μεθόδων ανταλλαγής δεδομένων.

2.3. ΤΕΧΝΙΚΕΣ ΣΥΝΕΡΓΑΣΙΑΣ ΕΦΑΡΜΟΓΩΝ

2.1.6. Η εξάπλωση των γραφικών διεπαφών χρήστη

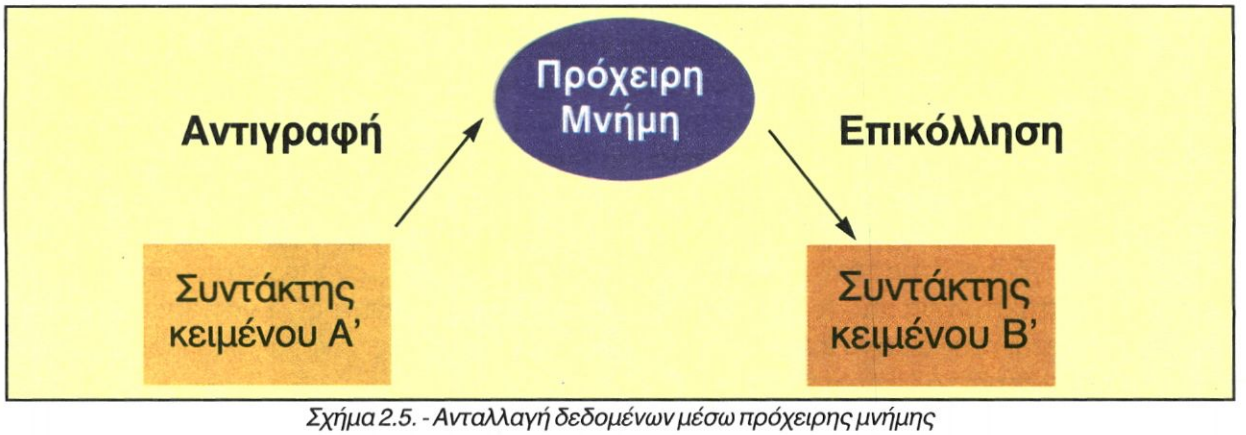
Από το 1984 και μετά, μια σημαντική καινοτομία στην επικοινωνία χρήστη – μηχανής εμφανίστηκε, αρχικά από την εταιρία Apple και τον υπολογιστή McIntosh και στα τέλη της δεκαετίας του '80 εμφανίστηκε και στους προσωπικούς υπολογιστές, καθώς και στους σταθμούς εργασίας UNIX. Στη θέση της γραμμής εντολών και της πληκτρολόγησης, ο χρήστης χειρίζεται τον υπολογιστή με τη βοήθεια μιας συσκευής κατάδειξης (συνήθως ποντίκι), μενού, γραφικά πλαίσια διαλόγου, κουμπιά, εικονίδια που προσομοιώνουν συμπεριφορά πραγματικών αντικειμένων (μπορούμε να τα μετακινήσουμε, τα πετούμε στο καλάθι αχρήστων για να τα διαγράψουμε κ.λ.π.) και γενικά ένα πλήθος γραφικών επινοήσεων, μέσω των οποίων ο χρήστης δίνει στον υπολογιστή εντολές.

Ο τρόπος αυτός επικοινωνίας χρήστη – μηχανής λέγεται **Γραφική Διεπαφή Χρήστη** *(Graphical User Interface – GUI)*. Τα πιο ευρέως διαδεδομένα παραδείγματα γραφικών διεπαφών χρήστη είναι οι διάφορες εκδόσεις των Windows της εταιρίας Microsoft, το λειτουργικό σύστημα MacOS της εταιρίας Apple και το "παραθυρικό σύστημα που ονομάζεται X", ή αλλιώς τα X Windows όπως για συντομία ονομάζονται και κυριαρχούν στους σταθμούς εργασίας με λειτουργικό σύστημα UNIX.

Τα X Windows βασίζονται σε ένα ερευνητικό πρόγραμμα του Τεχνολογικού Ινστιτούτου της Μασαχουσέτης και υποστηρίζονται από το Ίδρυμα Ανοιχτού Λογισμικού. Οι γραφικές διεπαφές χρήστη έκαναν το χειρισμό του υπολογιστή πολύ πιο εύκολο, ιδιαίτερα για τους αρχάριους χρήστες και έδωσαν μεγάλη ώθηση στην εξάπλωση των προσωπικών υπολογιστών, όχι μόνο στους χώρους δουλειάς, αλλά και στο σπίτι. Νέες ισχυρές εφαρμογές οργάνωσης γραφείου εμφανίστηκαν, προσφέροντας δυνατότητες στα όρια της επιτραπέζιας τυπογραφίας, συνδυασμένες με σχεσιακές βάσεις δεδομένων, λογιστικά φύλλα, λογισμικό οργάνωσης του χρόνου, λογισμικό για ζωγραφική κ.λ.π.

Παράλληλα με τις αλλαγές στον τρόπο προβολής της πληροφορίας και χειρισμού του υπολογιστή που εισήγαγαν, οι γραφικές διεπαφές προσέφεραν έναν απλό και λειτουργικό τρόπο για να ανταλλάσσουν δεδομένα οι εφαρμογές μεταξύ τους με τη βοήθεια απλά του ποντικιού, χωρίς να χρειάζεται να κατασκευαστούν ειδικά φίλτρα, αρχεία δέσμης εντολών του λειτουργικού συστήματος ή τροποποιήσεις των εφαρμογών.

Ο απλούστερος τρόπος για να συμβεί αυτό είναι με τη βοήθεια μιας **πρόχειρης μνήμης** *(clipboard)*. Μέσω αυτής αντιγράφουμε το κείμενο από έναν συντάκτη (editor) κειμένου σε έναν άλλο (**σχήμα 2.5**).



Σχήμα 2.5. - Ανταλλαγή δεδομένων μέσω πρόχειρης μνήμης

Μπορείτε να αναφέρετε από την προσωπική σας πείρα περιπτώσεις ανταλλαγής δεδομένων μεταξύ εφαρμογών στο περιβάλλον των Windows;

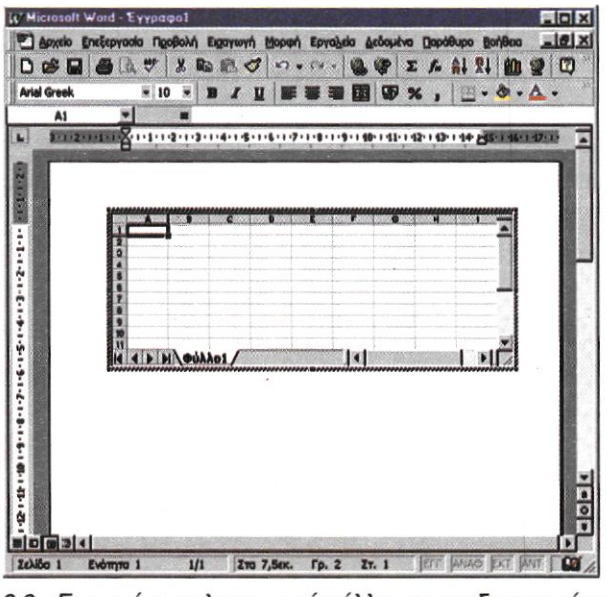
Πέρα όμως από την αντιγραφή απλού κειμένου, τα συστήματα που χρησιμοποιούν γραφικές διεπαφές χρήστη προσφέρουν πιο ισχυρές δυνατότητες ανταλλαγής δεδομένων. Αυτές καθορίζονται από κατάλληλα πρωτόκολλα, έτσι ώστε να εξασφαλίζεται ότι σε όλες τις εφαρμογές που λειτουργούν μέσω της συγκεκριμένες γραφικής διεπαφής (π.χ., Microsoft Windows) να μπορεί ο χρήστης να χειρίζεται την ανταλλαγή δεδομένων με τον ίδιο τρόπο (κοινά μενού, ονόματα εντολών, κινήσεις ποντικιού, συντομογραφίες πλήκτρων κ.λ.π.).

2.1.7. Σύγχρονες τεχνικές συνεργασίας εφαρμογών

Όπως αναφέραμε και στην προηγούμενη παράγραφο, με την εξάπλωση των γραφικών διεπαφών χρήστη εμφανίστηκαν νέες τεχνικές ανταλλαγής δεδομένων και επικοινωνίας εφαρμογών που προσέφεραν πολύ περισσότερες δυνατότητες από τις τεχνικές που χρησιμοποιούνταν μέχρι τότε.

Μία τέτοια τεχνική για ανταλλαγή δεδομένων και επικοινωνία εφαρμογών είναι το πρωτόκολλο **Σύνδεση και Ενσωμάτωση Αντικειμένων** *(Object Linking and Embedding – OLE)* που παρουσιάστηκε από την εταιρία Microsoft το 1991 και το 1993 το εξέλιξε στην έκδοση 2.0 που είναι η έκδοση που υποστηρίζουν οι σύγχρονες εφαρμογές για Windows. Η OLE ακολουθεί την αρχιτεκτονική πελάτη εξυπηρετητή. Η εφαρμογή "εξυπηρετητής" είναι η εφαρμογή που χρησιμοποιείτε για να δημιουργήσετε δεδομένα (π.χ., ένα λογιστικό φύλλο) και η εφαρμογή "πελάτης" είναι η εφαρμογή (π.χ., ένας επεξεργαστής κειμένου), στην οποία εισάγετε τα δεδομένα που δημιουργήσατε με την εφαρμογή "εξυπηρετητής".

Μέσω της OLE μπορούμε να συνδέσουμε ένα έγγραφο μιας εφαρμογής με ένα έγγραφο μιας άλλης εφαρμογής, έτσι ώστε οι πληροφορίες να μεταβιβάζονται αυτόματα. Για παράδειγμα, μπορούμε να συνδέσουμε μια αναφορά που δημιουργήθηκε με έναν επεξεργαστή κειμένου με έναν τιμοκατάλογο που δημιουργήθηκε με ένα λογιστικό φύλλο, έτσι ώστε κάθε φορά που θα αλλάζουν μέσω του λογιστικού φύλλου οι τιμές στον τιμοκατάλογο η αναφορά να ενημερώνεται αυτόματα. Κατά την ενσωμάτωση αντικειμένων, μπορείτε να ενσωματώσετε σε ένα αρχείο της εφαρμογής "πελάτη" ένα τμήμα ενός αρχείου της εφαρμογής "εξυπηρετητή" και με το ποντίκι να επεξεργαστείτε το αρχείο της εφαρμογής "εξυπηρετητή" μέσα από την εφαρμογή "πελάτη". Αυτό προϋποθέτει η εφαρμογή "εξυπηρετητής" να είναι εγκατεστημένη στον ίδιο υπολογιστή με την εφαρμογή "πελάτης". Στο στάδιο της επεξεργασίας η εφαρμογή "πελάτης" μεταμορφώνεται και αποκτά στοιχεία από το μενού της εφαρμογής "εξυπηρετητής". Ένα παράδειγμα αυτού απεικονίζεται στην **εικόνα 2.2** όπου εφαρμογή "εξυπηρετητής" είναι το Microsoft Excel και εφαρμογή "πελάτης" το Microsoft Word.



Εικόνα 2.2. - Ενσωμάτωση λογιστικού φύλλου σε επεξεργαστή κειμένου.

Μία άλλη δυνατότητα που προσφέρει το συγκεκριμένο πρωτόκολλο είναι ο αυτοματισμός OLE Αυτοματισμός. Δίνει τη δυνατότητα στο χρήστη να προσπελαύνει και να τροποποιεί αντικείμενα (π.χ., κείμενα, φύλλα εργασίας, εικόνες κ.λ.π.) μέσω γλωσσών υψηλού επιπέδου (π.χ., Visual Basic for Applications, Object Pascal κλπ). Για παράδειγμα μπορούμε να κατασκευάσουμε μια εφαρμογή σε Delphi και να χρησιμοποιήσουμε το Word για εκτύπωση αναφορών. Ο έλεγχος του Word μέσα από το πρόγραμμά μας θα γίνεται μέσω του αυτοματισμού OLE. Σε αυτόν βασίζεται επίσης η δυνατότητα προγραμματισμού με μακροεντολές των εφαρμογών αυτοματισμού γραφείου (σύνδεση λογιστικών φύλλων με βάσεις δεδομένων κ.λ.π.).

Παρόμοια λειτουργικότητα με τη σύνδεση μέσω OLE προσέφερε και η Δυναμική Ανταλλαγή Δεδομένων (Dynamic Data Exchange – DDE) που υλοποιήθηκε από την Microsoft στα Windows 3.0 το 1990. Το πρωτόκολλο αυτό όμως υπέφερε και από πολλά προβλήματα, το σημαντικότερο εκ των οποίων ήταν η σπατάλη μνήμης.

Ένα άλλο σύγχρονο και δημοφιλές πρωτόκολλο για την επικοινωνία εφαρμογών, ανεξάρτητα από τον τρόπο κατασκευής τους και τον υπολογιστή στον οποίο είναι εγκατεστημένες είναι η **Αρχιτεκτονική Διαμεσολάβησης για Αιτήματα Κοινών Αντικειμένων** *(Common Object Request Broker Architecture – CORBA).* Αυτή θεσπίστηκε από την ομάδα OMG (Object Manager Group), μια μη κερδοσκοπική εταιρία που ιδρύθηκε το 1989 από οκτώ μεγάλες εταιρίες (3Com Corporation, American Airlines, Canon, Inc., Data General, Hewlett-Packard, Philips Telecommunications N.V., Sun Microsystems and Unisys Corporation).

Τα πρωτόκολλα CORBA και COM είναι τα δύο επικρατέστερα για την ανταλλαγή δεδομένων και τη συνεργασία εφαρμογών σήμερα.

Η αρχιτεκτονική του CORBA είναι και αυτή πελάτη – εξυπηρετητή και σημαντικό ρόλο σε αυτή παίζει ο *Μεσίτης Αιτήσεων για Αντικείμενα (Object Request Broker – ORB).* Χρησιμοποιώντας την ενδιάμεση εφαρμογή ORB η εφαρμογή – πελάτης μπορεί να καλέσει μία μέθοδο (συνάρτη-

ση) από την εφαρμογή – εξυπηρετητή η οποία μπορεί να βρίσκεται είτε στον ίδιο υπολογιστή είτε σε κάποιον άλλο που συνδέεται μέσω δικτύου. Ο ORB δέχεται την κλήση και είναι υπεύθυνος να βρει μια εφαρμογή πελάτη που να μπορεί να ανταποκριθεί σε αυτήν την κλήση, να καλέσει την μέθοδο που ζήτησε ο πελάτης περνώντας τις κατάλληλες παραμέτρους και να προσφέρει το αποτέλεσμα της κλήσης στον πελάτη. Η εφαρμογή – πελάτης δε χρειάζεται να γνωρίζει που είναι εγκατεστημένη η εφαρμογή εξυπηρετητής, με ποια γλώσσα προγραμματισμού έχει κατασκευαστεί και με πιο λειτουργικό σύστημα λειτουργεί.

Ανακεφαλαίωση / περίληψη

Ο παραδοσιακός τρόπος ανταλλαγής δεδομένων μεταξύ των εφαρμογών λογισμικού είναι μέσω της ανταλλαγής αρχείων. Διαφορές όμως στη λογική οργάνωση των εγγραφών των αρχείων και η διαφορετική αρχιτεκτονική των επεξεργαστών συνήθως επιβάλλουν τη μετατροπή της μορφής ενός αρχείου για να μπορέσει να το διαβάσει μια εφαρμογή διαφορετική από εκείνη που το κατασκεύασε. Το λογισμικό που κάνει αυτή τη μετατροπή λέγεται φίλτρο.

Η εξάπλωση των τοπικών δικτύων και η αύξηση της υπολογιστικής ισχύος των προσωπικών υπολογιστών δημιούργησε τις προϋποθέσεις για την εξάπλωση των συστημάτων πελάτη – εξυπηρετητή. Ο εξυπηρετητής που συνήθως είναι ένας ισχυρός προσωπικός υπολογιστής, επεξεργάζεται τα δεδομένα ανταποκρινόμενος σε αιτήσεις των εφαρμογών που εκτελούνται στους υπολογιστές στις θέσεις εργασίας, οι οποίοι ονομάζονται και πελάτες (clients). Οι εξυπηρετητές μέσω του δικτύου στέλνουν στους πελάτες το αποτέλεσμα της επεξεργασίας των δεδομένων.

Τα συστήματα αυτά που απαρτίζονται από υπολογιστές διαφορετικής αρχιτεκτονικής και λειτουργικών συστημάτων στους οποίους εκτελούνται εφαρμογές διαφορετικών κατασκευαστών που συνεργάζονται μεταξύ τους λέγονται ανοιχτά συστήματα. Ένας από τους δυνατούς τρόπους για την επικοινωνία μιας εφαρμογής – πελάτη με ένα ΣΔΒΔ – εξυπηρετητή είναι μέσω του ODBC που προσφέρει έναν πρότυπο τρόπο επικοινωνίας, ανεξάρτητο από τον κατασκευαστή του ΣΔΒΔ.

Τέλος, τα σύγχρονα πρωτόκολλα ανταλλαγής δεδομένων για την ανταλλαγή δεδομένων και τη συνεργασία μεταξύ εφαρμογών είναι η Σύνδεση και Ενσωμάτωση Αντικειμένων (OLE) και η Αρχιτεκτονική Διαμεσολάβησης για Αιτήματα Κοινών Αντικειμένων (CORBA). Η πρώτη σχεδιάστηκε από την Microsoft και κυριαρχεί σε υπολογιστές με λειτουργικά συστήματα της ίδιας εταιρίας, ενώ η δεύτερη είναι διατυπωμένη από την κοινοπραξία OMG και δεν εξαρτάται από συγκεκριμένο λειτουργικό σύστημα.

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Τι είναι η αρχιτεκτονική πελάτη – εξυπηρετητή;
2. Το σύστημα στο σχολικό εργαστήριο θα μπορούσε να χαρακτηριστεί ανοιχτό; Προσπαθήστε να τεκμηριώσετε την απάντησή σας.
3. Τι είναι φίλτρο; Δώστε παραδείγματα χρήσης φίλτρων από εφαρμογές του σχολικού εργαστηρίου.
4. Τι είναι ο κώδικας ASCII;
5. Τι είναι ODBC; Καταγράψτε τους οδηγούς ODBC που είναι εγκατεστημένοι στους υπολογιστές του σχολικού εργαστηρίου.
6. Αναφέρατε παραδείγματα εφαρμογών από το σχολικό εργαστήριο που μπορούν να λειτουργήσουν είτε ως εξυπηρετητές είτε ως πελάτες σε μια ανταλλαγή δεδομένων OLE.
7. Σε τι διαφέρει η σύνδεση και ενσωμάτωση από τον αυτοματισμό OLE;

Γλωσσάριο - Ευρετήριο όρων

Αρχιτεκτονική πελάτη-εξυπηρετητή – Μοντέλο λειτουργίας υπολογιστικών συστημάτων στο οποίο μια εφαρμογή (που ονομάζεται εξυπηρετητής), επεξεργάζεται δεδομένα για λογαριασμό μιας ομάδας εφαρμογών (που ονομάζονται πελάτες), στους οποίους προσφέρει τα αποτελέσματα της επεξεργασίας. Η επικοινωνία εξυπηρετητή – πελατών γίνεται συνήθως μέσω κάποιου πρωτοκόλλου δικτύου [50].

ASCII – Κώδικας που αντιστοιχεί αλφαριθμητικά ψηφία (γράμματα, αριθμούς, σημεία στίξης κ.λ.π.) της αγγλικής γλώσσας σε ακέραιους αριθμούς από το 0 έως το 127. Γλώσσες διαφορετικές από την αγγλική χρησιμοποιούν τους αριθμούς από 128 έως 255 για να κωδικοποιήσουν τα δικά τους ψηφία ή σύμβολα [46].

EBCDIC – Κώδικας αντιστοίχησης αλφαριθμητικών ψηφίων με ακέραιους αριθμούς, που χρησιμοποιείται στους μεγάλους κεντρικούς υπολογιστές (mainframes) της IBM [46].

OLE – Πρωτόκολλο ανταλλαγής δεδομένων και συνεργασίας εφαρμογών, που υποστηρίζεται από την οικογένεια λειτουργικών συστημάτων Windows της εταιρίας Microsoft [50].

CORBA - Πρωτόκολλο ανταλλαγής δεδομένων και συνεργασίας εφαρμογών, ανεξάρτητα λειτουργικών συστημάτων. Υποστηρίζεται από μια πληθώρα εκδόσεων του UNIX, αλλά εφαρμογές του υπάρχουν και σε εφαρμογές σχεδιασμένες για την οικογένεια λειτουργικών Microsoft Windows [51].

Βιβλιογραφία

Bennet David, "Visual C++ 5.0 Developer's Guide", SAMS publishing, 1997.
Mattison Rob., "Directions In Database Management Systems: Selection and Implementation", Computer Technology Research Corp., 1991.
Raymond Eric., "The New Hacker's Dictionary", Second Edition, MIT Press, (1994).
Venditto Gus., "Ο Οδηγός των Windows 3.1", Εκδόσεις Γκιούρδας, 1992.

Χρήσιμες πηγές πληροφοριών αποτελούν οι ακόλουθες τοποθεσίες στο διαδίκτυο:
http://www.microsoft.com/data/odbc Για πληροφορίες σχετικές με ODBC.
http://www.omg.org Η βασική σελίδα του OMG απ' όπου μπορείτε να πάρετε πολλές πληροφορίες για το CORBA.
http://www.msdn.microsoft.com/developer Πληροφορίες για την τεχνολογία OLE και γενικά για κάθε τεχνολογία ή εργαλείο προγραμματισμού της Microsoft.
http://www.ieee.org Πληροφορίες για το πρότυπο POSIX.
http://www.osf.org Η βασική σελίδα του Open Systems Foundation. Μπορείτε να βρείτε πληροφορίες για X Windows, τεχνολογία δικτύων και ανταλλαγής δεδομένων.

ΚΕΦΑΛΑΙΟ 3

ΜΕΛΛΟΝΤΙΚΕΣ ΤΑΣΕΙΣ

ΠΕΡΙΕΧΟΜΕΝΑ

3.1. Οι τρέχουσες εξελίξεις
3.2. Μελλοντικές εξελίξεις

ΕΙΣΑΓΩΓΗ

Με την εξάπλωση του διαδίκτυου *(internet)* οι εφαρμογές, είτε διαχείρισης δεδομένων είτε άλλων κατηγοριών, στράφηκαν στον Παγκόσμιο Ιστό, όσον αφορά στη διεπαφή με το χρήστη και στα συστήματα πολλών βαθμίδων, όσον αφορά στην αρχιτεκτονική των εξυπηρετητών. Παράλληλα η αρχιτεκτονική του διαδικτύου χρησιμοποιείται και για κλειστά ενδοεπιχειρησιακά δίκτυα *(intranets)*, που φιλοξενούν ολόκληρο το Πληροφοριακό Σύστημα Διοίκησης μιας εταιρίας.

- Τι είναι ο Παγκόσμιος Ιστός και πως στάθηκε δυνατό να προβληθούν μέσω αυτού πληροφορίες από βάσεις δεδομένων;
- Θα ενοποιηθεί το γραφικό περιβάλλον εργασίας των προσωπικών υπολογιστών με τον Παγκόσμιο Ιστό;
- Ποιες αναμένεται να είναι οι εξελίξεις του άμεσου μέλλοντος στις εφαρμογές λογισμικού;

Στο παρόν κεφάλαιο θα δούμε αναλυτικότερα αυτές τις εξελίξεις, ρίχνοντας παράλληλα μια σύντομη ματιά σε ορισμένες νέες τεχνικές που αναμένεται να βρούνε μεγαλύτερη εφαρμογή στο άμεσο μέλλον.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΕΠΙΘΥΜΗΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

- Να εντοπίσει ο μαθητής την επιρροή του Παγκόσμιου Ιστού στις εφαρμογές λογισμικού και να γνωρίσουμε τις τεχνικές που χρησιμοποιούνται για δημιουργία εφαρμογών στον Παγκόσμιο Ιστό και για τη σύνδεση με βάσεις δεδομένων.
- Να γνωρίσει τι σημαίνει σύστημα πολλών βαθμίδων και ποια η σημασία του.
- Να γνωρίσει τις νέες εξελίξεις στον Παγκόσμιο Ιστό σε σχέση με τη γλώσσα κατασκευής ιστοσελίδων και τις τεχνικές δημιουργίας τρισδιάστατων ιστοσελίδων.
- Να αποκτήσει μια στοιχειώδη γνώση για τις αντικειμενοστρεφείς βάσεις δεδομένων.
- Να μπορει να σχολιάσει και να κρίνει τις τρέχουσες εξελίξεις και τις διαφαινόμενες τάσεις στην εξέλιξη των εφαρμογών λογισμικού.

3.1. ΟΙ ΤΡΕΧΟΥΣΕΣ ΕΞΕΛΙΞΕΙΣ

3.1.1. Οι εφαρμογές λογισμικού στον Παγκόσμιο Ιστό

Την αφορμή για την ταχύτατη εξάπλωση του διαδικτύου έξω από τα ακαδημαϊκά ιδρύματα έδωσε η ανακάλυψη στις αρχές της δεκαετίας του 1990 από τον Tim Berners – Lee στο Ευρωπαϊκό Εργαστήριο Φυσικής Σωματιδίων (CERN), του Παγκόσμιου Ιστού (World Wide Web).

```
<HTML>
<HEAD>
<TITLE>Ministry of Education And Religious Affairs</TITLE>
</HEAD>

<frameset cols="184, *" frameborder=no border=0>
 <frame src="menubar. html" marginwidth=0 marginheight=0 scrolling="no" name="menu" noresize >
 <frameset rows="*">
  <frame src="home. shtml" marginwidth=0 marginheight=0 scrolling="auto" name="mainpage" resize >
  </frameset>
 </frameset>
<NOFRAMES>
<BODY>
You need a browser capable of displaying frames to see this page : (
</NOFRAMES>
</BODY>
</HTML>
```

Εικόνα 3.1. - Παράδειγμα σύνταξης HTML

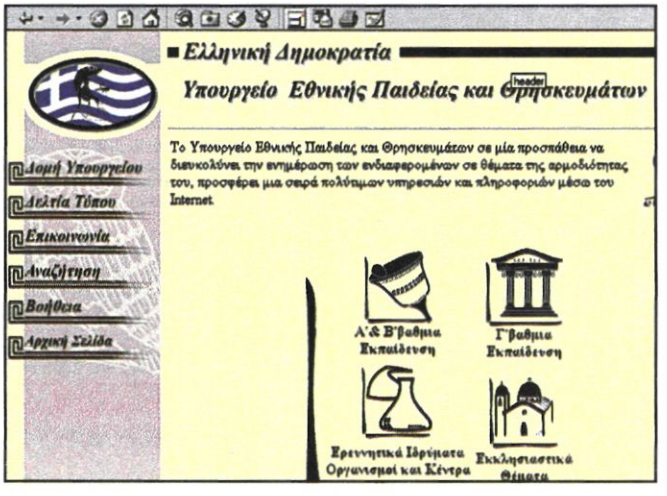
Η βασική ιδέα του Παγκόσμιου Ιστού είναι η χρησιμοποίηση του **υπερκειμένου** για την προβολή πληροφοριών μέσω του **διαδίκτυου** *(internet)*. Το υπερκείμενο είναι μια τεχνική προβολής κειμένου μέσω του ηλεκτρονικού υπολογιστή. Το κείμενο είναι οργανωμένο σε σελίδες και ενδεχομένως σε κεφάλαια, όπως σε ένα συμβατικό βιβλίο. Με τη διαφορά ότι υπάρχουν ειδικά επισημασμένες λέξεις με τις οποίες ο χρήστης μπορεί, πιέζοντας με το ποντίκι πάνω τους, να μεταφερθεί σε μια άλλη θέση του βιβλίου, ή ακόμα και σε ένα άλλο βιβλίο, στο οποίο αυτή η λέξη θα αναπτύσσεται καλύτερα. Για παράδειγμα, αν το βιβλίο που διαβάζετε ήταν σε μορφή υπερκειμένου, θα μπορούσατε πιέζοντας με το ποντίκι στη λέξη *διαδίκτυο* να μεταφερθείτε σε μία σελίδα ενός βιβλίου που αναφέρεται σε τεχνολογία δικτύων και περιγράφει την αρχιτεκτονική του διαδίκτυου.

Μπορούμε να φανταστούμε τον παγκόσμιο ιστό ως μια τεράστια βιβλιοθήκη, όπου όλα τα βιβλία είναι σε μορφή υπερκειμένου. Στο υπερκείμενο εκτός από κείμενο ενσωματώνονται και διάφορα άλλα αντικείμενα (εικόνες, αρχεία βίντεο, μουσικής, εφαρμογές λογισμικού κ.λ.π.).

Αν έχετε ήδη πλοηγηθεί στον Παγκόσμιο Ιστό, μπορείτε να συνοψίσετε τι είδους χρήσεις του κάνατε;

Τα έγγραφα που προβάλλονται στον Παγκόσμιο Ιστό συντάσσονται αυτή τη στιγμή στη **Γλώσσα Χαρακτηρισμού Υπερκειμένου *(Hypertext Markup Language – HTMLHTML)***, αργότερα ίσως αντικατασταθεί από άλλη (π.χ., την XML που θα δούμε στη συνέχεια). Η HTML δεν είναι γλώσσα προγραμματισμού, αλλά γλώσσα περιγραφής και μορφοποίησης εγγράφων. Βασίζεται στη χρήση ετικετών *(tags)* με τις οποίες ορίζονται παράγραφοι, μορφοποιούνται επικεφα-

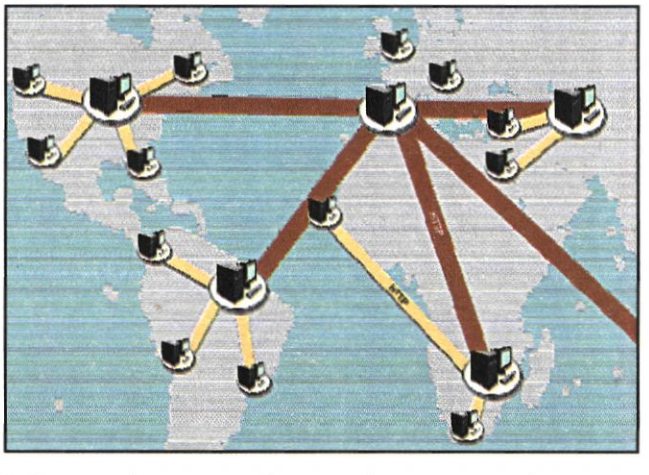
λίδες, σχηματίζονται λίστες κ.λ.π. Απόσπασμα HTML από την ιστοσελίδα του Υπουργείου Παιδείας φαίνεται στην **εικόνα 3.1** Οι ετικέτες περικλείονται σε < >.



Εικόνα 3.2. - Ιστοσελίδα όπως ερμηνεύεται και προβάλλεται από την εφαρμογή φυλλομέτρησης σελίδων.

Η HTML είναι η γλώσσα σύνταξης των ιστοσελίδων.

Το κείμενο που έχει συνταχθεί σε HTML ερμηνεύεται και προβάλλεται στην οθόνη του υπολογιστή από εφαρμογές **φυλλομέτρησης** ιστοσελίδων (*internet browsers* – φυλλομετρητές του διαδίκτυου). Το αποτέλεσμα φαίνεται στην **εικόνα 3.2** Οι ιστοσελίδες είναι αποθηκευμένες σε συστήματα υλικού και λογισμικού που λέγονται **εξυπηρετητές του ιστού** *(web servers)*. Το σύνολο των ιστοσελίδων που καλύπτουν ένα συγκεκριμένο θέμα (π.χ., παρουσίαση μιας εταιρίας) μαζί με τον εξυπηρετητή του ιστού μέσω του οποίου διαθέτονται στο διαδίκτυο λέγεται *τοποθεσία του ιστού ή ιστόπεδο (web site)*. Ένας εξυπηρετητής του ιστού μπορεί να φιλοξενεί περισσότερα από ένα ιστόπεδα.



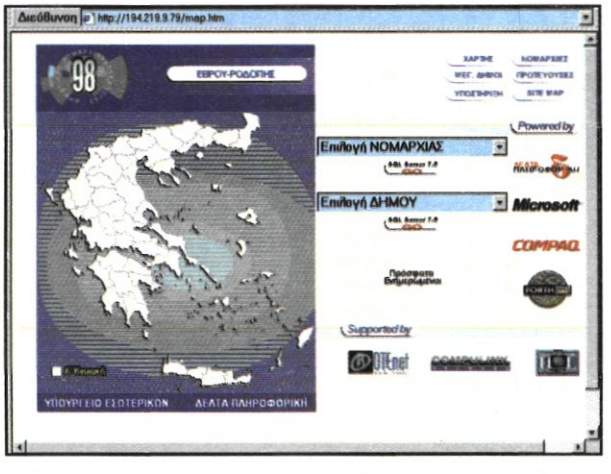
Σχήμα 3.1. - Απλουστευμένη απεικόνιση του Παγκόσμιου Ιστού (εικόνα από εκπαιδευτικό CD-ROM της Microsoft).

Η αρχιτεκτονική του Παγκόσμιου Ιστού είναι αρχιτεκτονική πελάτη – εξυπηρετητή. Πελάτης είναι η εφαρμογή φυλλομέτρησης των ιστοσελίδων και εξυπηρετητής είναι ο εξυπηρετητής του ιστού. Μεταξύ τους συνδέονται μέσω του πρωτοκόλλου δικτύου TCP/IP. Για την ακρίβεια, το TCP/IP είναι ένα σύνολο ιεραρχημένων πρωτοκόλλων (σε επάλληλα στρώματα). Η συνομιλία του πελάτη με τον εξυπηρετητή για το σχηματισμό του παγκόσμιου ιστού βασίζεται στο πρωτόκολλο μεταφοράς υπερκειμένου HTTP *(Hypertext Transfer Protocol)*, το οποίο αποτελεί ένα πρωτόκολλο επιπέδου εφαρμογών, από τα ανώτερα στρώματα του TCP/IP. Μέσω του HTTP γίνεται η μεταφορά αρχείων (υπερκειμένου, γραφικών, ήχων κ.λ.π.) από τον εξυπηρετητή στον πελάτη. Στο **σχήμα 3.1** απεικονίζεται μια απλοποιημένη δομή του Παγκόσμιου Ιστού με τους εξυπηρετητές να συνδέονται μεταξύ τους με κόκκινες λωρίδες και τους πελάτες να συνδέονται με τους εξυπηρετητές με κίτρινες λωρίδες.

Ένα από τα ισχυρά χαρακτηριστικά του παγκόσμιου ιστού είναι η **διαλογική** του φύση. Ο χρήστης δεν ξεφυλλίζει μόνο ιστοσελίδες, αλλά μπορεί να αναζητήσει πληροφορίες σε βάσεις δεδομένων, να κάνει παραγγελίες αγαθών, να επεξεργαστεί δεδομένα κ.λ.π. Στην **εικόνα 3.3** απεικονίζεται μια βάση δεδομένων που είναι διαθέσιμη στον Παγκόσμιο Ιστό.

Στην τοποθεσία **http://www.terraserver.com** μπορείτε να δείτε μια τεράστια βάση δεδομένων με φωτογραφίες της Γης από δορυφόρο.

Επίσης από την τοποθεσία http://www.ypes.gr μπορείτε να μεταβείτε στη βάση δεδομένων με τα αποτελέσματα των Δημοτικών Εκλογών 1998.



Εικόνα 3.3. - Βάση δεδομένων διαθέσιμη στον Παγκόσμιο Ιστό.

Η διαλογική φύση εξασφαλίζεται με τεχνικές που επιτρέπουν:

- Την επικοινωνία και ανταλλαγή δεδομένων των εξυπηρετητών ιστοσελίδων με προγράμματα και βάσεις δεδομένων. Τα προγράμματα μπορεί να είναι μεταγλωττισμένα (π.χ., από μεταγλωττιστές C, Pascal κ.λ.π.), αρχεία εντολών δέσμης του λειτουργικού συστήματος (π.χ., Unix shell scripts), ή γραμμένα σε ειδική γλώσσα (π.χ., Javascript), τα οποία διερμηνεύει και παράγει ιστοσελίδες ο ίδιος ο εξυπηρετητής ιστοσελίδων. Συνηθισμένο πρωτόκολλο για την επικοινωνία του εξυπηρετητή με βάσεις δεδομένων είναι το ODBC (βλ. 2.1.2).
- Την ενσωμάτωση στις ιστοσελίδες προγραμμάτων, είτε με τη μορφή πηγαίου κώδικα ειδικής γλώσσας, ο οποίος διερμηνεύεται από το φυλλομετρητή (π.χ., Javascript), είτε με τη μορφή ήδη μεταγλωττισμένων (ή σχεδόν μεταγλωττισμένων στην περίπτωση της Java) αντικειμένων.

Από την τοποθεσία **http://www.developer.netscape.com** μπορείτε να βρείτε εγχειρίδια και άρθρα για τη Javascript και άλλες τεχνολογίες του Παγκόσμιου Ιστού.

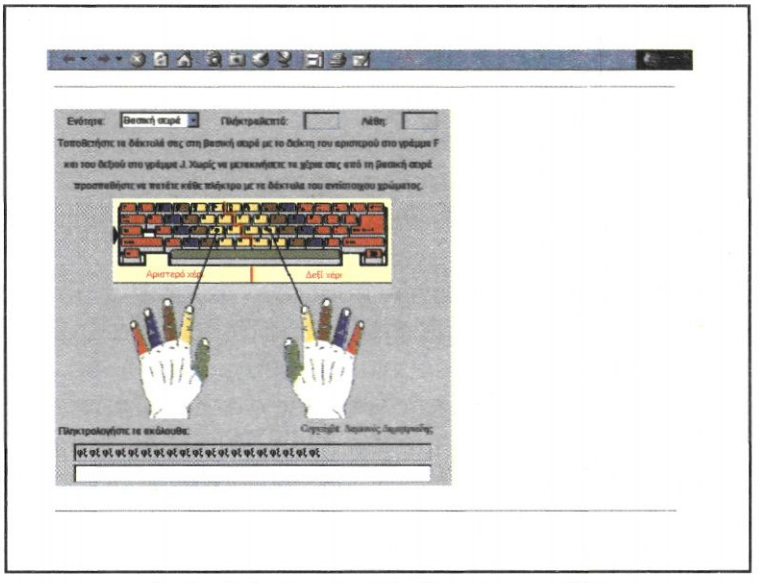
Επίσης στην τοποθεσία **http://www.javasoft.com** μπορείτε να βρείτε εφαρμογές της java, εγχειρίδια και ηλεκτρονική τεκμηρίωση, καθώς και εργαλεία για ανάπτυξη εφαρμογών Java (Java Developer Kits).

Στην πρώτη από τις παραπάνω κατηγορίες ανήκουν οι ακόλουθες τεχνολογίες:

- Κοινή διεπαφή πύλης *(CGI – common gateway interface)*. Είναι ένας πρότυπος τρόπος με τον οποίο ένας εξυπηρετητής του ιστού περνά το αίτημα του χρήστη σε μία εφαρμογή, αποδέχεται το αποτέλεσμα της επεξεργασίας και το επιστρέφει μέσω του ιστού στο χρήστη. Η CGI εξασφαλίζει την επιτυχία της επικοινωνίας ανεξάρτητα από το λειτουργικό σύστημα του εξυπηρετητή (Windows, MacIntosh, Unix, OS/390 κ.λ.π.) και τη γλώσσα προγραμματισμού με την οποία έχει φτιαχτεί η εφαρμογή (π.χ., C, Pascal, Unix shell scripts κ.λ.π.).
- Γλώσσες προγραμματισμού διερμηνευόμενες στον εξυπηρετητή. Αυτές είναι γλώσσες που εμφανίστηκαν από κατασκευαστές εφαρμογών φυλλομέτρησης ιστοσελίδων τα τελευταία χρόνια. Αναλαμβάνει ο εξυπηρετητής να τις διερμηνεύσει και να στείλει στον πελάτη ιστοσελίδες HTML ως αποτέλεσμα. Παραδείγματα τέτοιας τεχνικής είναι οι Active Server Pages της Microsoft, που εμπεριέχουν προγράμματα γραμμένα σε VBScript ή Jscript. Επιπρόσθετα, πολλές εφαρμογές υποστηρίζουν σύνδεση με βάση δεδομένων (συνήθως μέσω ODBC), με συνέπεια να μπορούν να κατασκευάσουν φόρμες διαχείρισης των δεδομένων που να προβάλλονται μέσω φυλλομετρητών ιστοσελίδων.

Στη δεύτερη ανήκουν οι παρακάτω:

- Γλώσσες προγραμματισμού διερμηνευόμενες στον πελάτη. Είναι πανομοιότυπες με τις προηγούμενες, μόνο που ενσωματώνονται ως πηγαίος κώδικας στην ιστοσελίδα και η εφαρμογή πελάτης τις περνά από διερμηνέα *(interpreter)* και τις εκτελεί τοπικά. Μέσω αυτών μπορούμε να αυξήσουμε τα διαλογικά χαρακτηριστικά των ιστοσελίδων (π.χ., να αλλάζουμε το σχήμα ενός κουμπιού μόλις ο δείκτης του ποντικιού περάσει από επάνω κ.λ.π.). Χαρακτηριστικότερο παράδειγμα τέτοιας γλώσσας είναι η JavaScript, ή JScript.
- Εφαρμογές Java. Πρόκειται για αντικειμενοστρεφή γλώσσα προγραμματισμού κατασκευασμένη από την εταιρία Sun και εξαιρετικά διαδεδομένη. Μπορεί να κατασκευάσει τόσο ολοκληρωμένες εφαρμογές, όσο και μικρές εφαρμογές *(java applets)* που ενσωματώνονται μέσα σε ιστοσελίδες. Το μεγάλο της πλεονέκτημα είναι ότι οι εφαρμογές που κατασκευάζονται με αυτήν μπορούν να εκτελεστούν σε μια πληθώρα υπολογιστών και λειτουργικών συστημάτων (Unix, Windows, MacOS, OS/2 κ.λ.π.). Αυτό μπορεί να γίνει γιατί όταν κατασκευάζουμε μια εφαρμογή Java το παραγόμενο προϊόν δεν είναι κανονικός εκτελέσιμος κώδικας μηχανής, αλλά ένας ενδιάμεσος "σχεδόν" εκτελέσιμος κώδικας *(bytecode)*. Για να μπορέσει ένας υπολογιστής να εκτελέσει τον ενδιάμεσο κώδικα πρέπει να είναι εφοδιασμένος με ένα ειδικό λογισμικό, που ονομάζεται εικονική μηχανή Java *(JVM – Java Virtual Machine)*. Αυτή άλλοτε προσφέρεται μαζί με το λειτουργικό σύστημα και άλλοτε προσφέρεται μαζί με το λογισμικό φυλλομέτρησης ιστοσελίδων. Στην **εικόνα 3.4** φαίνεται ένα java applet όπως φαίνεται σε μια ιστοσελίδα. Πρόσφατα διατυπώθηκε από τη θυγατρική εταιρία της Sun, Javasoft το πρωτόκολλο JDBC μέσω του οποίου μπορεί μια εφαρμογή ή ένα java applet να επικοινωνεί με βάση δεδομένων. Η βάση δεδομένων μπορεί να βρίσκεται στον ίδιο υπολογιστή με τον εξυπηρετητή ιστού ή σε άλλον υπολογιστή συνδεδεμένο με τον πρώτο μέσω δικτύου. Επίσης πολλές υλοποιήσεις του JDBC προσφέρουν τη δυνατότητα σύνδεσης με το διαχειριστή οδηγών ODBC, έτσι ώστε να μπορεί να συνδεθεί με όλα τα ΣΔΒΔ που υποστηρίζουν το τελευταίο πρωτόκολλο.
- Εξαρτήματα ActiveX *(ActiveX components)*. Είναι η εναλλακτική λύση της Microsoft απέναντι στα java applets. Ουσιαστικά είναι προγράμματα που για να εκτελεστούν χρειάζεται να φιλοξενηθούν στο παράθυρο άλλων εφαρμογών (π.χ., αναγνώστες ιστοσελίδων, εφαρμογές κατασκευασμένες από γλώσσες 4ης γενιάς κ.λ.π.). Τα ActiveX βασίζονται στην ίδια αρχιτεκτονική με το πρωτόκολλο OLE που είδαμε στο δεύτερο κεφάλαιο. Η αρχιτεκτονική αυτή ονομάζεται Μοντέλο Αντικειμένων Εξαρτημάτων *(COM – Component Object Model)*. Απέναντι στα java applets έχουν το πλεονέκτημα της πιο γρήγορης εκτέλεσης (μια που τα ActiveX είναι δυαδικός κώδικας μηχανής) και τα μειονεκτήματα ότι εκτελούνται μόνο στα λειτουργικά συστήματα της Microsoft, καθώς και ότι προσφέρουν μειωμένη προστασία (σε σχέση με τα java applets) του υπολογιστή από εξαρτήματα κατασκευασμένα με σκοπό να προκαλέσουν ζημιά.



Εικόνα 3.4. - Java applet μέσα σε ιστοσελίδα.

3.1.2. Συστήματα διαχείρισης δεδομένων 3 ή περισσότερων βαθμίδων

Με την ανάπτυξη των πρωτοκόλλων απομακρυσμένης συνεργασίας και των εργαλείων γρήγορης ανάπτυξης των εφαρμογών δημιουργήθηκαν οι προϋποθέσεις για επέκταση του μοντέλου πελάτη – εξυπηρετητή, έτσι ώστε να μπορεί να διαχειριστεί μεγάλες ποσότητες δεδομένων πιο αποτελεσματικά.

Αυτό γίνεται διαχωρίζοντας το σύστημα σε τρεις ή περισσότερες βαθμίδες (**σχήμα 3.2**). Κάθε βαθμίδα μπορεί να αποτελείται από έναν ή περισσότερους υπολογιστές συνδεδεμένους μέσω δικτύου. Μέσω δικτύου επίσης, επικοινωνούν και οι βαθμίδες μεταξύ τους.

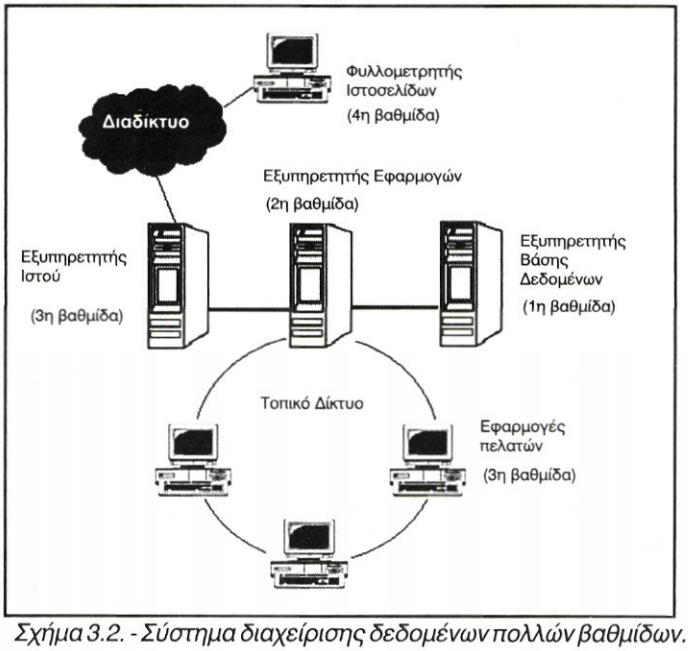
Η **πρώτη** βαθμίδα αποτελείται από τους υπολογιστές που φιλοξενούν τα **συστήματα διαχείρισης βάσεων δεδομένων** (ΣΔΒΔ). Συνήθως για αυξημένη ασφάλεια δεν επιτρέπουν τη άμεση σύνδεση των χρηστών με τα ΣΔΒΔ, παρά μόνο μέσω της δεύτερης βαθμίδας.

Η **δεύτερη** βαθμίδα αποτελείται από τις εφαρμογές που περιέχουν τη λεγόμενη **"λογική της επιχείρησης"** *(business logic)*, δηλαδή τον κώδικα βασικής επεξεργασίας των δεδομένων. Ο υπολογιστής (ή οι υπολογιστές) που φιλοξενούν αυτές τις εφαρμογές ονομάζεται *εξυπηρετητής εφαρμογών (application server)*. Αυτός λειτουργεί σαν εξυπηρετητής της τρίτης βαθμίδας και σαν πελάτης της πρώτης. Παραλαμβάνει τα δεδομένα από τα ΣΔΒΔ της πρώτης βαθμίδας, τα επεξεργάζεται και προσφέρει τα αποτελέσματα στις εφαρμογές της τρίτης βαθμίδας. Οι εξυπηρετητές εφαρμογών είναι σχεδιασμένοι ώστε να παρέχουν τις υπηρεσίες 24 ώρες το εικοσιτετράωρο, 7 ημέρες την εβδομάδα, ενώ σε περίπτωση πτώσης του συστήματος αναλαμβάνουν την ανασύσταση των πληροφοριών. Οι εξυπηρετητές εφαρμογών δεν αποθηκεύουν δεδομένα, αλλά συνήθως είναι επιφορτισμένοι με τον έλεγχο αυθεντικότητας των χρηστών. Οι

Ο **TerraServer**, η βάση εικόνων από δορυφόρο, βασίζεται σε ένα σύστημα τριών βαθμίδων. Συνοπτική περιγραφή μπορείτε να βρείτε στην τοποθεσία **http://www.microsoft.com/backoffice/scalability/tserver.htm**

εφαρμογές της δεύτερης βαθμίδας βασίζονται σε μια ειδική κατηγορία λογισμικού, το **ενδιάμεσο λογισμικό** *(middleware)*, το οποίο είναι βιβλιοθήκες συναρτήσεων και μεθόδων, με τη βοήθεια των οποίων επιτυγχάνεται η επικοινωνία των εφαρμογών και η ανταλλαγή των δεδομένων. Το ενδιάμεσο λογισμικό βασίζεται στα πρωτόκολλα ανταλλαγής δεδομένων και συνεργασίας των εφαρμογών που είδαμε στο 2° κεφάλαιο, όπως το CORBA ή το DCOM (μια επέκταση του πρωτοκόλλου COM στο οποίο βασίζεται το OLE και τα εξαρτήματα ActiveX).

Η **τρίτη** βαθμίδα είναι η βαθμίδα των **πελατών** σύμφωνα με την παραδοσιακή αρχιτεκτονική πελάτη – εξυπηρετητή (βλ. 2° κεφάλαιο), με τη διαφορά ότι το μεγαλύτερο μέρος από τον κώδικα επεξεργασίας των δεδομένων μεταφέρθηκε στη δεύτερη βαθμίδα. Οι εφαρμογές της τρίτης βαθμίδας είναι συνήθως προσωπικοί υπολογιστές ή σταθμοί εργασίας που λειτουργούν μέσω μιας ποικιλίας λειτουργικών συστημάτων (Unix, Windows, MacOS κ.λ.π.), και στους οποίους τρέχουν εφαρμογές για την επικοινωνία με το χρήστη μέσω γραφικών διεπαφών ή μέσω τερματικών χαρακτήρων. Με λίγα λόγια αυτές οι εφαρμογές προσφέρουν τα μενού, τα πλαίσια διαλόγου και γενικά όλα τα αντικείμενα μέσω των οποίων ο χρήστης επικοινωνεί με το σύστημα. Σε πολλές περιπτώσεις αυτή η επικοινωνία γίνεται μέσω ιστοσελίδων και το μόνο που χρειάζεται να έχουν οι πελάτες είναι ο κατάλληλος φυλλομετρητής. Σε αυτή την περίπτωση ο εξυπηρετητής ιστού που φιλοξενεί (ή κατασκευάζει με τη βοήθεια διερμηνέα, βλ. 3.1.1) τις ιστοσελίδες αποτελεί την τρίτη βαθμίδα και οι υπολογιστές με τους φυλλομετρητές αποτελούν μια τέταρτη βαθμίδα.



Σχήμα 3.2. - Σύστημα διαχείρισης δεδομένων πολλών βαθμίδων.

Τέτοιου είδους εφαρμογές πολλών βαθμίδων χρησιμοποιούνται και σε εταιρικά δίκτυα που έχουν παρόμοια αρχιτεκτονική με το διαδίκτυο και τον Παγκόσμιο Ιστό, αλλά δεν επιτρέπουν

την προσπέλαση σε χρήστες εκτός του εταιρικού δικτύου. Αυτού του είδους τα εταιρικά δίκτυα ονομάζονται "intranets".

Τα συστήματα τριών η περισσότερων βαθμίδων εξασφαλίζουν μεγαλύτερη επεκτασιμότητα (μπορεί να αυξηθεί το μέγεθος οποιασδήποτε βαθμίδας χωρίς να επηρεαστούν οι υπόλοιπες), μεγαλύτερη ικανότητα επεξεργασίας δεδομένων και μεγαλύτερη ασφάλεια. Όλα αυτά με τίμημα το μεγαλύτερο κόστος σε υλικό και λογισμικό.

3.2. ΜΕΛΛΟΝΤΙΚΕΣ ΕΞΕΛΙΞΕΙΣ

3.2.1. Αντικειμενοστρεφείς βάσεις δεδομένων

Με τη διεύρυνση των πεδίων εφαρμογών του λογισμικού (π.χ., στη βιομηχανία, ψηφιοποίηση και διαχείριση οπτικοακουστικού υλικού κ.λ.π.), παρουσιάστηκαν κάποια προβλήματα στη διαχείριση σύνθετων δεδομένων που ξέφευγαν από τις παραδοσιακές κατηγορίες των αριθμητικών ή αλφαριθμητικών (π.χ., λέξεις, προτάσεις κ.λ.π.) δεδομένων. Παραδείγματα τέτοιων σύνθετων δεδομένων ήταν π.χ., οι ψηφιοποιημένες εικόνες και ήχοι, σύνθετα ηλεκτρονικά ή μηχανολογικά ψηφιακά σχέδια κ.λ.π.

Μια αρχική λύση που δόθηκε από τους κατασκευαστές σχεσιακών ΣΔΒΔ ήταν η αποθήκευσή τους σε πεδία **μεγάλων δυαδικών αντικειμένων** *(Binary Large Objects - BLOB)* και η χρήση βοηθητικών πεδίων που είτε περιέγραφαν το περιεχόμενο των BLOB, είτε περιείχαν κωδικούς χρήσιμους για τη συσχέτισή τους με άλλα δεδομένα.

Τα αντικειμενοστρεφή ΣΔΒΔ βασίστηκαν στην εμπειρία που αποκτήθηκε από τη χρήση των αντικειμενοστρεφών γλωσσών προγραμματισμού (π.χ. Smaltalk, C++, Object Pascal κ.λ.π.)

Μια εναλλακτική λύση όμως εμφανίστηκε γύρω στο 1992 με την εμφάνιση των **αντικειμενοστρεφών** ΣΔΒΔ. Σύμφωνα με αυτήν τα δεδομένα αποθηκεύονται με την μορφή **αντικειμένων**. Αυτά τα αντικείμενα περιέχουν όχι μόνο την τιμή του δεδομένου, αλλά και τις μεθόδους με τις οποίες αυτό αλληλεπιδρά με το περιβάλλον του. Αυτή η αλληλεπίδραση περιλαμβάνει, για παράδειγμα, την εκτέλεση αριθμητικών ή λογικών πράξεων, την απόδοση τιμής, την εμφάνιση επιμέρους ιδιοτήτων του αντικείμενου κ.λ.π.

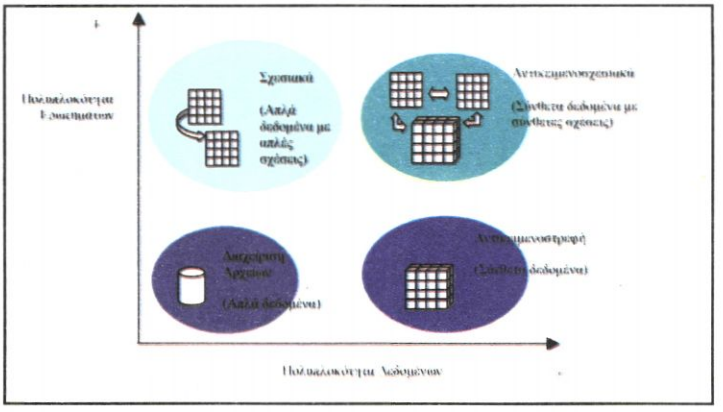
Με βάση τα όσα είπαμε παραπάνω μπορούμε να συμπεράνουμε ότι η βάση δεδομένων ενός αντικειμενοστρεφούς ΣΔΒΔ δεν περιέχει μόνο τα δεδομένα αλλά καταχωρεί μεθόδους και συναρτήσεις επεξεργασίας της καταχωρημένης πληροφορίας. Και τα σχεσιακά συστήματα προσπαθούν με την μορφή των αποθηκευμένων διαδικασιών *(stored procedures)* να ενσωματώσουν ανάλογες πληροφορίες στη βάση δεδομένων, αλλά στις αντικειμενοστρεφείς αυτό γίνεται με πιο ολοκληρωμένο και συνεπή τρόπο. Με απλά λόγια, τα δεδομένα που διαχειρίζεται η αντικειμενοστρεφής βάση δεδομένων δεν είναι "παθητικά" όπως στα κλασικά αρχεία και είναι πιο "ενεργητικά" από τα σχεσιακά ΣΔΒΔ.

Οι αντικειμενοστρεφείς βάσεις δεδομένων χρησιμοποιούνται σε μικρή έως σήμερα κλίμακα,

κυρίως σε εφαρμογές σχεδίασης (CAD) και κατασκευής (CAM) με Η/Υ, σε διαχείριση πολυμέσων και γενικά σε διαχείριση σύνθετων δεδομένων. Τα περισσότερα όμως σημερινά αντικειμενοστρεφή ΣΔΒΔ υπολείπονται σε διαθεσιμότητα εργαλείων και ολοκλήρωση σε σχέση με τα ώριμα σχεσιακά ΣΔΒΔ της αγοράς και είναι δύσχρηστα στον προγραμματισμό (συνήθως απαιτούν C++ ή Java). Παραδείγματα αντικειμενοστρεφών ΣΔΒΔ της αγοράς είναι η Jasmine της Computer Associates, η ObjectStore της Object Design, η ODMG της O2 Technology η Versant της ομώνυμης εταιρίας κ.λ.π.

Η λύση που τείνει να επικρατήσει μεταξύ των κατασκευαστών ΣΔΒΔ είναι ο συνδυασμός των χαρακτηριστικών των σχεσιακών με τα αντικειμενοστρεφή ΣΔΒΔ με την εμφάνιση των *αντικειμενοσχεσιακών* ΣΔΒΔ. Τέτοια συστήματα έχουν εμφανιστεί από τους περισσότερους μεγάλους κατασκευαστές σχεσιακών ΣΔΒΔ (π.χ Oracle, IBM, Informix, Sybase κ.λ.π.).

Το **σχήμα 3.3** απεικονίζει τις ανάγκες στις οποίες ανταποκρίνονται καλύτερα οι διάφορες λύσεις που υπάρχουν σήμερα για συστήματα λογισμικού διαχείρισης δεδομένων.



Σχήμα 3.3. - Ανάγκες στις οποίες ανταποκρίνονται καλύτερα οι λύσεις διαχείρισης δεδομένων.

3.2.2. Εξελίξεις βασισμένες στον Παγκόσμιο Ιστό

Η εξάπλωση του Παγκόσμιου Ιστού είτε στο διαδίκτυο είτε στα intranets και η εξάπλωση του ηλεκτρονικού εμπορίου έχουν οδηγήσει όλο και περισσότερους κατασκευαστές λογισμικού να σχεδιάζουν τα προϊόντα τους ενσωματώνοντας πελάτες με την μορφή ιστοσελίδων. Μάλιστα οι νέες εκδόσεις των Windows προσπαθούν να ενοποιήσουν την εικόνα που έχει ο χρήστης για τον τοπικό υπολογιστή του και για το διαδίκτυο, προσφέροντας ένα ενιαίο περιβάλλον εργασίας.

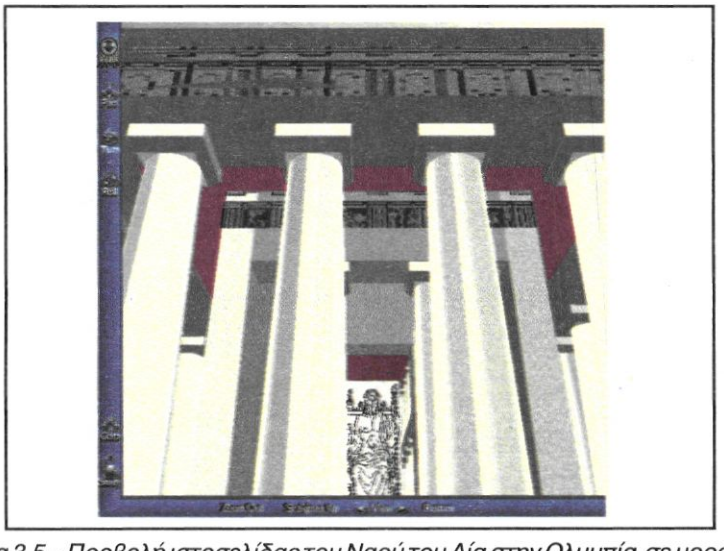
Στο πλαίσιο αυτό έχει κάνει πρόσφατα την εμφάνισή της η γλώσσα **Επεκτάσιμη Γλώσσα Χαρακτηρισμού, ή XML** *(eXtensible Markup Language)*. Αυτή είναι μια γλώσσα περιγραφής των συ-

στατικών και της δομής των κειμένων και όχι γλώσσα προγραμματισμού, που υιοθετήθηκε πρόσφατα από την κοινοπραξία W3C (www. consortium), που εποπτεύει τον Παγκόσμιο Ιστό.

Από την τοποθεσία της Κοινοπραξίας WWW, **http://www.w3c.org** μπορείτε να βρείτε περισσότερες πληροφορίες για την XML, την HTML και θέματα σχετικά με την τεχνολογία του Παγκόσμιου Ιστού.

Η πιο τυπική εφαρμογή της θα είναι η αναπαράσταση δεδομένων και η ανταλλαγή τους μεταξύ πελατών και εξυπηρετητών. Η XML αναμένεται να χρησιμοποιηθεί σε εφαρμογές ηλεκτρονικού εμπορίου, μια που μπορεί να δώσει μια τυποποιημένη μορφή για να ανταλλάσσεται η πληροφορία μεταξύ οργανισμών και εταιριών με ασύμβατες εφαρμογές. Μέχρι στιγμής δεν υπάρχουν πολλοί διαθέσιμοι φυλλομετρητές που να υποστηρίζουν XML, αλλά σύντομα αναμένεται η νέα γενιά φυλλομετρητών που θα μπορούν να διαβάζουν άμεσα XML.

Μία άλλη τεχνολογία που αναπτύχθηκε γύρω στο 1995 από την εταιρία Silicon Graphics δίνει τη δυνατότητα να δημιουργήσει κανείς τρισδιάστατες τοποθεσίες στον Παγκόσμιο Ιστό. Η τεχνολογία αυτή ονομάζεται **Γλώσσα Μοντελοποίησης Εικονικής Πραγματικότητας** ή **VRML** *(Virtual Reality Modeling Language)*. Η VRML (**εικόνα 3.5**) είναι μια γλώσσα που επιτρέπει την περιγραφή, σύνθεση και προβολή τρισδιάστατων αντικειμένων, μπορεί να τα θέσει σε κίνηση και να ενσωματώσει επίσης και ήχους. Μολονότι οι δυνατότητες που προσφέρει είναι μεγάλες, εντούτοις δεν έχει εξαπλωθεί ιδιαίτερα λόγω κυρίως των μεγάλων της απαιτήσεων σε εύρος ζώνης των τηλεπικοινωνιακών καναλιών.



Εικόνα 3.5. - Προβολή ιστοσελίδας του Ναού του Δία στην Ολυμπία σε μορφή VRML (παραγωγή Ιδρύματος Μείζονος Ελληνισμού.

Τα τελευταία χρόνια όμως η τεχνολογία OpenGL (στην οποία βασίζεται η VRML) έχει ενσωματωθεί και στο λειτουργικό σύστημα Windows (95, 98 ή NT) και κυκλοφορούν σε χαμηλές τιμές ειδικές κάρτες επιτάχυνσης για προβολή εικόνων OpenGL και VRML από προσωπικούς υπολογιστές. Επίσης οι περισσότεροι σύγχρονοι φυλλομετρητές υποστηρίζουν VRML, που σημαίνει ότι με την εξέλιξη της τεχνολογίας και την μείωση του κόστους, μπορεί η εξάπλωσή της να είναι σημαντικά μεγαλύτερη.

Ανακεφαλαίωση / περίληψη

Η δημιουργία του Παγκόσμιου Ιστού έδωσε τεράστια ώθηση στη χρήση του διαδίκτυου, αλλά και καθιέρωσε νέα πρότυπα στη σχεδίαση των γραφικών διεπαφών των εφαρμογών. Η πληροφορίες μέσα στον παγκόσμιο ιστό προβάλλονται με την μορφή ιστοσελίδων, που είναι έγγραφα υπερκειμένου κατασκευασμένα προς το παρόν με τη γλώσσα HTML. Αυτά τα έγγραφα μπορούν να αποκτήσουν διαλογικά χαρακτηριστικά (π.χ., ενσωμάτωση κουμπιών και πεδίων καταχώρησης κειμένου, ανταπόκριση σε κινήσεις του ποντικιού κ.λ.π.) με τη βοήθεια διερμηνευομένων γλωσσών προγραμματισμού υψηλού επιπέδου, όπως η JavaScript και η VBScript. Η διαλογική φύση των εγγράφων επαυξάνεται με τεχνικές που επιτρέπουν την επικοινωνία και ανταλλαγή δεδομένων των εξυπηρετητών ιστοσελίδων με προγράμματα και βάσεις δεδομένων.

Για να ανταποκριθούν στις αυξημένες απαιτήσεις διαχείρισης μεγάλων βάσεων δεδομένων και ασφαλούς προβολής στους στο διαδίκτυο, τα παραδοσιακά συστήματα πελάτη – εξυπηρετητή εξελίχθηκαν σε συστήματα τριών ή περισσότερων βαθμίδων. Σε αυτά η τρίτη βαθμίδα αναλαμβάνει την παρουσίαση στο χρήστη, είτε αυτή είναι φόρμες σε οθόνες χαρακτήρων, είτε παραθυρικές φόρμες, είτε ιστοσελίδες που φιλοξενούνται ή κατασκευάζονται από εξυπηρετητές ιστού. Η δεύτερη βαθμίδα περιέχει τον κώδικα βασικής επεξεργασίας των δεδομένων. Οι εξυπηρετητές της δεύτερης βαθμίδας λέγονται εξυπηρετητές εφαρμογών. Η πρώτη βαθμίδα αποτελείται από τα ΣΔΒΔ.

Με την εξάπλωση των πολυμέσων και των σύνθετων δεδομένων (ηλεκτρονικά σχέδια κ.λ.π.) οι κατασκευαστές σχεσιακών ΣΔΒΔ εμπλούτισαν τα χαρακτηριστικά των συστημάτων τους με αντικειμενοστρεφείς δυνατότητες, παράγοντας τα αντικειμενοσχεσιακά ΣΔΒΔ, ή δημιούργησαν νέα ΣΔΒΔ τα αντικειμενοστρεφή. Σε αυτά, τα δεδομένα αποθηκεύονται με τη μορφή αντικειμένων τα οποία δεν περιέχουν μόνο την τιμή των δεδομένων, αλλά και μεθόδους και συναρτήσεις επεξεργασίας της και αλληλεπίδρασής τους με άλλα αντικείμενα.

Μια νέα γλώσσα κατασκευής ιστοσελίδων, η XML, προβλέπεται να χρησιμοποιηθεί ευρέως στο μέλλον, μια που μπορεί να δώσει μια τυποποιημένη μορφή για να ανταλλάσσεται η πληροφορία μεταξύ οργανισμών και εταιριών με ασύμβατες εφαρμογές.

Ενδιαφέρον παρουσιάζει επίσης και η γλώσσα VRML για τη δημιουργία τρισδιάστατων ιστοσελίδων με δυνατότητα κίνησης στο χώρο. Η διάδοσή της προς το παρόν δεν είναι ιδιαίτερα μεγάλη, λόγω του μεγέθους των σελίδων που κατασκευάζει και των μεγάλων της απαιτήσεων σε τηλεπικοινωνιακή υποδομή.

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Σε τι διαφέρει το υπερκείμενο από το απλό κείμενο;
2. Είναι η HTML μια γλώσσα προγραμματισμού;
3. Από τις ακόλουθες γλώσσες προγραμματισμού, κάποιες μπορεί να ενσωματωθούν ως πηγαίος κώδικας μέσα σε ιστοσελίδες και να εκτελεστούν από το διερμηνέα του φυλλομετρητή. Επιλέξτε τις κατάλληλες.

❑ Fortran	❑ Javascript	❑ C++
❑ VBScript	❑ Java	❑ Pascal

4. Επισκεφθείτε, αν υπάρχει η δυνατότητα στο σχολικό εργαστήριο, τις ιστοσελίδες του Προγράμματος Περσέας (http://www.perseus.tufts.edu) και παρατηρήστε τις τεχνικές που χρησιμοποιεί για διαχείριση δεδομένων και για επικοινωνία με το χρήστη.
5. Περιγράψτε το μοντέλο τριών βαθμίδων στα συστήματα διαχείρισης και επεξεργασίας δεδομένων. Ποια τα πλεονεκτήματά και τα μειονεκτήματά του;
6. Τι είναι ενδιάμεσο λογισμικό;
7. Γιατί η XML θεωρείται κατάλληλη για εφαρμογές ηλεκτρονικού εμπορίου;
8. Εγκαταστήστε στο σχολικό εργαστήριο πρόσθετο λογισμικό για απεικόνιση VRML αν χρειάζεστε (διαθέσιμο στο (http://tucows.forthnet.gr) και επισκεφθείτε τις ιστοσελίδες του Ιδρύματος Μείζονος Ελληνισμού (http://www.fhw.gr) και παρατηρήστε τα VRML μοντέλα που υπάρχουν εκεί.

Γλωσσάριο - Ευρετήριο όρων

αντικειμενοστρεφείς βάσεις δεδομένων *(object oriented DBMS)* - Βάσεις δεδομένων που περιέχουν όχι μόνο τα δεδομένα και την περιγραφή της δομής του κάθε αρχείου, αλλά και τις μεθόδους και συναρτήσεις επεξεργασίας της καταχωρημένης πληροφορίας. Χρησιμοποιούνται σε μικρή έως σήμερα κλίμακα, κυρίως σε εφαρμογές σχεδίασης (CAD) και παραγωγής (CAM) με Η/Υ, σε διαχείριση πολυμέσων και γενικά σε διαχείριση σύνθετων δεδομένων [63].

εξυπηρετητής ιστού *(web server)* - Λογισμικό που δέχεται αιτήσεις από τους φυλλομετρητές ιστοσελίδων και αναλαμβάνει να τους τροφοδοτήσει με ιστοσελίδες ή άλλα δεδομένα (π.χ., αρχεία για μεταφορά). Το πρωτόκολλο επικοινωνίας του εξυπηρετητή ιστού με τους φυλλομετρητές, ονομάζεται HTTP [58].

κοινή διεπαφή πύλης *(Common Gateway Interface CGI)* – Ένας πρότυπος τρόπος, με τον οποίο ένας εξυπηρετητής του Παγκόσμιου Ιστού περνά το αίτημα του χρήστη σε μια εφαρμογή, αποδέχεται το αποτέλεσμα της επεξεργασίας και το επιστρέφει μέσω του Ιστού στο χρήστη [60].

πρότυπο (μοντέλο) αντικειμένων εξαρτημάτων *(COM)* - Πρωτόκολλο επικοινωνίας εφαρμογών και ανταλλαγής δεδομένων της εταιρίας Microsoft. Αποτελεί τη βάση των τεχνολογιών OLE (για σύνδεση και ενσωμάτωση αντικειμένων), ActiveX (για την ενσωμάτωση ενεργών

εξαρτημάτων σε ιστοσελίδες) και DCOM (για κατασκευή εξυπηρετητών εφαρμογών) [60].
συστήματα διαχείρισης δεδομένων τριών βαθμίδων *(three tier)* – Επέκταση του προτύπου "πελάτη – εξυπηρετητή" με την προσθήκη μιας ενδιάμεσης βαθμίδας μεταξύ των πελατών και του εξυπηρετητή βάσης δεδομένων. Αυτή η ενδιάμεση βαθμίδα ονομάζεται εξυπηρετητής εφαρμογών και περιέχει το βασικό κώδικα επεξεργασίας των δεδομένων [61].
υπερκείμενο *(hypertext)* - Ιστός πληροφοριών κειμένου, στον οποίο μπορεί να περιλαμβάνονται λέξεις, δράσεις ή εικόνες που αποτελούν συνδέσμους *(Links)* προς ένα άλλο σημείο του ίδιου ή άλλου εγγράφου [57].
HTML *(Hyper Text Markup Language)* - Γλώσσα περιγραφής και μορφοποίησης εγγράφων. Σε αυτήν συντάσσονται σήμερα οι ιστοσελίδες του Παγκόσμιου Ιστού. Βασίζεται στη χρήση ετικετών *(tags)* με τις οποίες ορίζονται παράγραφοι, μορφοποιούνται επικεφαλίδες, σχηματίζονται λίστες κ.λ.π. [57].
Java - Αντικειμενοστρεφής γλώσσα προγραμματισμού κατασκευασμένη από την εταιρία Sun. Μπορεί να κατασκευάσει τόσο ολοκληρωμένες εφαρμογές, όσο και μικρές εφαρμογές *(java applets)*, που ενσωματώνονται μέσα σε ιστοσελίδες. Το μεγάλο της πλεονέκτημα είναι ότι οι εφαρμογές που κατασκευάζονται με αυτήν μπορούν να εκτελεστούν σε μια πληθώρα υπολογιστών και λειτουργικών συστημάτων (Unix, Windows, MacOS, OS/2 κ.λ.π.) [59].
Javascript – Γλώσσα προγραμματισμού οι εφαρμογές της οποίας εκτελούνται μέσω διερμηνέα (*interpreter*). Μπορεί να ενσωματωθεί σε ιστοσελίδες και να διερμηνευθεί από το φυλλομετρητή, ή να διερμηνευθεί από τον εξυπηρετητή ιστού [59].
JDBC *(Java Data Base Modeling Language)* – Πρωτόκολλο για σύνδεση εφαρμογών Java με βάσεις δεδομένων [60].
VRML *(Virtual Reality Modelling Language)* – Γλώσσα σχηματισμού τρισδιάστατων εικόνων με δυνατότητα κίνησης και αλληλεπίδρασης με το χρήστη. Οι εικόνες αυτές μπορούν να προβάλλονται μέσω του Παγκόσμιου Ιστού [65].
VBScript *(Visual Basic Script)* - Γλώσσα προγραμματισμού οι εφαρμογές της οποίας εκτελούνται μέσω διερμηνέα (*interpreter*). Μπορεί να ενσωματωθεί σε ιστοσελίδες και να διερμηνευθεί από το φυλλομετρητή, ή να διερμηνευθεί από τον εξυπηρετητή ιστού [60].
XML (Extensible Markup Language) - Γλώσσα περιγραφής και μορφοποίησης εγγράφων, αποτελεί το διάδοχο της HTML. Κατάλληλη για εφαρμογές ηλεκτρονικού εμπορίου, γιατί μπορεί να δώσει μια τυποποιημένη μορφή για να ανταλλάσσεται η πληροφορία μεταξύ οργανισμών και εταιριών με ασύμβατες εφαρμογές [64].

Βιβλιογραφία

Vossen G., "Data Models, Database Languages and Database Management Systems", Addison Wesley 1991.

ΕΝΟΤΗΤΑ Β'

ΔΙΕΡΕΥΝΩ - ΣΥΣΧΕΤΙΖΩ

ΚΕΦΑΛΑΙΟ 4

ΚΑΤΑΝΟΗΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΚΑΙ ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

4.1. Ορισμός και κατανόηση του προβλήματος
4.2. Ορισμοί – Βασικές έννοιες συστημάτων
4.3. Το πρόβλημα της συγκέντρωσης και επεξεργασίας δεδομένων
4.4. Εκτίμηση των απαιτήσεων

ΕΙΣΑΓΩΓΗ

Εάν σε μια δημόσια υπηρεσία (π.χ. που εκδίδει πιστοποιητικά γέννησης) παρατηρήσουμε μεγάλες ουρές, το πιθανότερο που θα σκεφτούμε είναι ότι δεν επαρκούν οι υπάλληλοι για το δεδομένο όγκο εργασίας. Θα σκεφτούμε, δηλαδή ότι το πρόβλημα είναι η έλλειψη υπαλλήλων που θα έπρεπε να ασχολούνται με τη συγκεκριμένη εργασία. Μάλιστα πολλοί θα αναρωτηθούν: "γιατί δεν χρησιμοποιούν και άλλους υπαλλήλους, ώστε να μη ταλαιπωρείται το κοινό;" δίνοντας έτσι και την προφανή λύση στο πρόβλημα που δημιουργείται.

- Είναι όμως βέβαιο ότι αυτό είναι το πραγματικό πρόβλημα και όχι κάτι άλλο που κρύβεται πίσω από τα φαινόμενα;
- Πόσο θα μπορούσαμε να επιμείνουμε στην άποψή μας εάν κάποιος άλλος ισχυριζόταν ότι το πραγματικό πρόβλημα δεν είναι η έλλειψη υπαλλήλων, αλλά οι χρονοβόρες διαδικασίες που προβλέπονται για τη συγκέντρωση των στοιχείων που απαιτούνται για την έκδοση ενός πιστοποιητικού;
- Η εάν κάποιος τρίτος έλεγε πως το πρόβλημα είναι η κακή οργάνωση του συστήματος αρχειθέτησης και αναζήτησης των απαιτούμενων πληροφοριών;

Γίνεται συνεπώς φανερό ότι, πριν αρχίσει η διαδικασία σχεδίασης και ανάπτυξης μιας εφαρμογής λογισμικού για την αντιμετώπιση κάποιου προβλήματος θα πρέπει να προσδιοριστεί με ακρίβεια και σαφήνεια το πρόβλημα, καθώς και το τι θα πρέπει να θεωρηθεί σαν λύση του. Αντικείμενο αυτού του κεφαλαίου είναι η παρουσίαση των πιο συνηθισμένων τεχνικών και μεθόδων που χρησιμοποιούνται για τη συγκέντρωση και την επεξεργασία των πληροφοριών, που επιτρέπουν το σωστό καθορισμό ενός προβλήματος και τον προσδιορισμό των απαιτήσεων μιας εφαρμογής (ή ενός συστήματος γενικότερα), που θα σχεδιασθεί και θα υλοποιηθεί για την επίλυσή του.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ - ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να διακρίνει τη διαφορά μεταξύ προβλήματος και συμπτώματος.
2. Να χρησιμοποιεί την τεχνική της ανάλυσης ενός προβλήματος σε υποπροβλήματα και να σχεδιάζει επιμέρους λύσεις.
3. Να χρησιμοποιεί τις βασικές έννοιες των συστημάτων προκειμένου να προσεγγίζει πολύπλοκα προβλήματα.
4. Να κατασκευάζει ερωτηματολόγια και να διεξάγει επιτυχημένες συνεντεύξεις, προκειμένου να συγκεντρώνει τα δεδομένα ανάλυσης ενός συστήματος.
5. Να εκτιμά τις απαιτήσεις ανάπτυξης μιας εφαρμογής.

4.1. ΚΑΘΟΡΙΣΜΟΣ ΚΑΙ ΚΑΤΑΝΟΗΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Ο καθορισμός ενός προβλήματος απαιτεί να προσδιορίσουμε ποιο ακριβώς είναι το πρόβλημα και τι θα πρέπει να θεωρηθεί ως λύση του.

Όταν οι άνθρωποι μιλούν για ένα πρόβλημα, δεν είναι καθόλου βέβαιο ότι έχουν στο μυαλό τους το ίδιο πράγμα. Είναι συνεπώς αναγκαίο, όταν μια ομάδα ανθρώπων αναλαμβάνει να λύσει κάποιο πρόβλημα, να συμφωνήσει εξ αρχής στο ποιο ακριβώς είναι το πρόβλημα και τι θα πρέπει να θεωρηθεί σαν λύση του. Με άλλα λόγια θα πρέπει να "καθορίσει" το πρόβλημα. Επίσης θα πρέπει να συμφωνήσει στο πως πρέπει να εργαστεί (δηλαδή με ποια μέθοδο), προκειμένου να σχεδιάσει και στη συνέχεια να υλοποιήσει τη λύση του.

Για το πώς πρέπει να εργαζόμαστε προκειμένου να καθορίσουμε και στη συνέχεια να επιλύσουμε ένα πρόβλημα σχεδίασης και υλοποίησης μιας εφαρμογής λογισμικού έχουν προταθεί διάφορα μεθοδολογικά πρότυπα (μοντέλα). Τα βασικά (κοινά) μεθοδολογικά βήματα που προτείνουν τα πρότυπα αυτά ή οι διάφορες παραλλαγές τους, απεικονίζονται στο **σχήμα 4.1.** της επόμενης σελίδας.

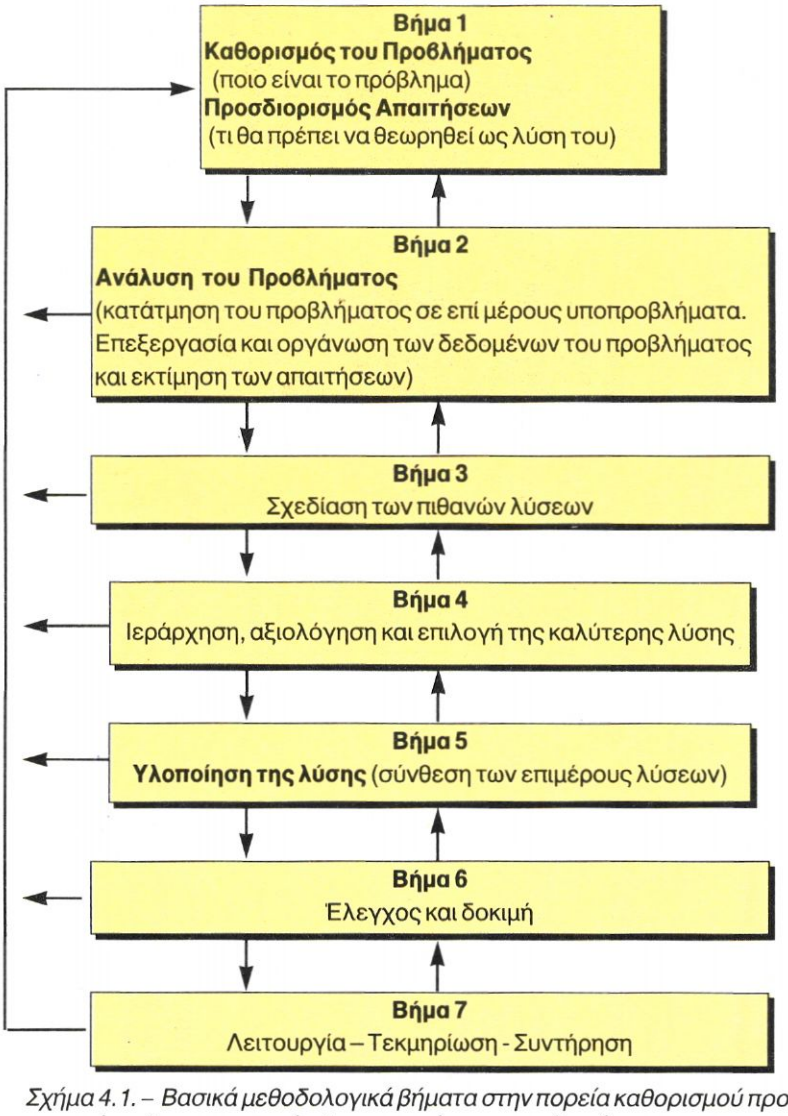
Τα βέλη στο σχήμα 4.1, ανάλογα με τη φορά τους, δείχνουν τον τρόπο μετάβασης από ένα βήμα σε άλλο. Όπως φαίνεται, η όλη πορεία είναι αμφίδρομη, αφού όπως θα εξηγηθεί παρακάτω, μετά την ολοκλήρωση των εργασιών κάθε βήματος, υπάρχει ανάγκη να επανεξεταστεί κάτω από άλλο πρίσμα, κάποιο ή κάποια από τα προηγούμενα βήματα. Κάθε ένα από τα βήματα που απεικονίζονται στο σχήμα 4.1 εξηγείται και αναλύεται διεξοδικά στα επόμενα κεφάλαια του παρόντος βιβλίου.

Ο καθορισμός του προβλήματος, που αποτελεί το πρώτο βήμα στην παραπάνω διαδικασία, συνίσταται στην αναζήτηση και στη μελέτη των επί μέρους στοιχείων, που θα μας επιτρέψει να εντοπίσουμε πού ακριβώς βρίσκεται το πρόβλημα, σε αντιδιαστολή με το πού αρχικά **φαίνεται** να είναι.

Κατ' αρχήν αναλύουμε λογικά το πρόβλημα, χωρίζοντάς το σε υποπροβλήματα, το κάθε ένα από τα οποία θα αντιμετωπισθεί χωριστά. Στη συνέχεια, από τις λύσεις των υποπροβλημάτων συνθέτουμε τη λύση του προβλήματος. Για να βοηθηθούμε στη λύση των προβλημάτων χρησιμοποιούμε **εργαλεία λογισμικού** καθώς και διάφορες **τεχνικές** και **μεθόδους**. Τα εργαλεία λογισμικού είναι προγράμματα, τα οποία χρησιμοποιούμε για να αναπτύξουμε την εφαρμογή μας, όπως ένας τεχνίτης χρησιμοποιεί τα δικά του εργαλεία για να αντιμετωπίσει κάποιο πρακτικό πρόβλημα. Τυπικό παράδειγμα τέτοιων εργαλείων είναι οι γλώσσες προγραμματισμού. Τεχνικές και μέθοδοι είναι κάποιες τυπικές διαδικασίες που ακολουθούμε για να επιλύσουμε κάποιο πρόβλημα. Οι διαδικασίες αυτές δεν χρησιμοποιούνται μόνο για την αντιμετώπιση συγκεκριμένου προβλήματος, αλλά και για κατηγορίες παρόμοιων προβλημάτων. Βασίζονται στην προγενέστερη εμπειρία που έχει αποκτηθεί από την επίλυση τέτοιων προβλημάτων και συνήθως είναι επιστημονικά τεκμηριωμένες. Στα παραπάνω θα πρέπει να προστεθεί και η πληθώρα γνώσεων και εμπειρίας, που διατίθεται σήμερα από το διαδίκτυο *(internet)*, γιατί σε πολλές περιπτώσεις είναι δυνατό να βρεθούν αυτούσιες λύσεις, ή έστω σημαντική βοήθεια σε

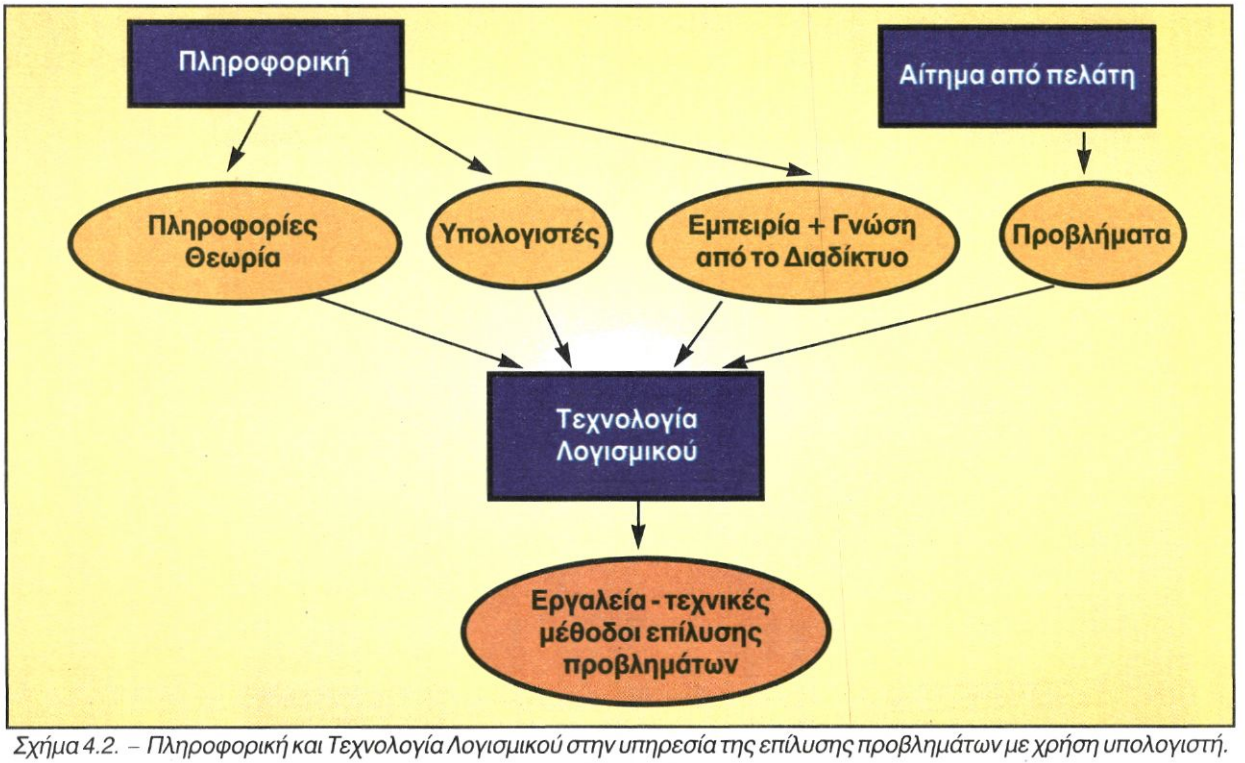
πολλές κατηγορίες προβλημάτων. Είναι συνεπώς δυνατό να κερδίσουμε χρόνο αξιοποιώντας την εμπειρία ή τις γνώσεις αυτές, για να λύσουμε ταχύτερα και το δικό μας πρόβλημα. Η συμπύκνωση και η επιστημονική θεμελίωση της γνώσης και της εμπειρίας που έχει αποκτηθεί μέσα από την αντιμετώπιση προβλημάτων σχετικών με την ανάπτυξη λογισμικού αποτελεί αντικείμενο ιδιαίτερου κλάδου της πληροφορικής, που ονομάζεται **"Τεχνολογία Λογισμικού"**. Η Τεχνολογία Λογισμικού αποσκοπεί στην ανάπτυξη μεθόδων και τεχνικών για την κατασκευή ποιοτικού λογισμικού.

Το πρώτο βήμα για την ανάπτυξη μιας εφαρμογής λογισμικού είναι να προσδιοριστεί με ακρίβεια και σαφήνεια το πρόβλημα στο οποίο θα δίνει λύση η εφαρμογή



Σχήμα 4.1. – Βασικά μεθοδολογικά βήματα στην πορεία καθορισμού προβλημάτων και ανάπτυξης εφαρμογών λογισμικού για την επίλυσή τους.
(Σημείωση: Για την εξήγηση των εννοιών "υποστήριξη – τεκμηρίωση" βλ. αναλυτικά στο κεφάλαιο 14).

Το **σχήμα 4.2.** μας δίνει μια εικόνα του πώς η Τεχνολογία Λογισμικού αναπτύσσει ή χρησιμοποιεί όλα τα απαραίτητα βοηθήματα, που μπορούν να αξιοποιηθούν για την επίλυση ενός προβλήματος.



Σχήμα 4.2. – Πληροφορική και Τεχνολογία Λογισμικού στην υπηρεσία της επίλυσης προβλημάτων με χρήση υπολογιστή.

Παράδειγμα:

Για να γίνει περισσότερο κατανοητός ο τρόπος με τον οποίο το πρόβλημα διαχωρίζεται σε υποπροβλήματα θα δοθεί ένα παράδειγμα. Ας υποτεθεί ότι το σχολείο μας διαθέτει τον κατάλληλο εξοπλισμό και ότι θέλουμε να δημιουργήσουμε ορισμένες ιστοσελίδες *(web pages)* στο διαδίκτυο, ώστε να μπορούν να χρησιμοποιούνται από οποιονδήποτε ενδιαφερόμενο, αφενός για γνωριμία με τις δραστηριότητες που αναπτύσσει η μαθητική κοινότητα και αφετέρου για δικούς μας εκπαιδευτικούς σκοπούς.

Κατ' αρχήν εντοπίζουμε στη διατύπωση του πιο πάνω προβλήματος τις λέξεις κλειδιά **ιστοσελίδες** και **διαδίκτυο**. Ο στόχος μας είναι να δημιουργήσουμε τις ιστοσελίδες με τις πληροφορίες και τη μορφή που θέλουμε να έχουν και να τις εμφανίσουμε με κάποιο τρόπο στο διαδίκτυο.

Τα δύο βασικά υποπροβλήματα, στα οποία μπορούμε να χωρίσουμε το κυρίως πρόβλημα είναι τα εξής:

Υποπρόβλημα 1: Η επιλογή τρόπου και μέσου για να παρουσιάσουμε τις ιστοσελίδες

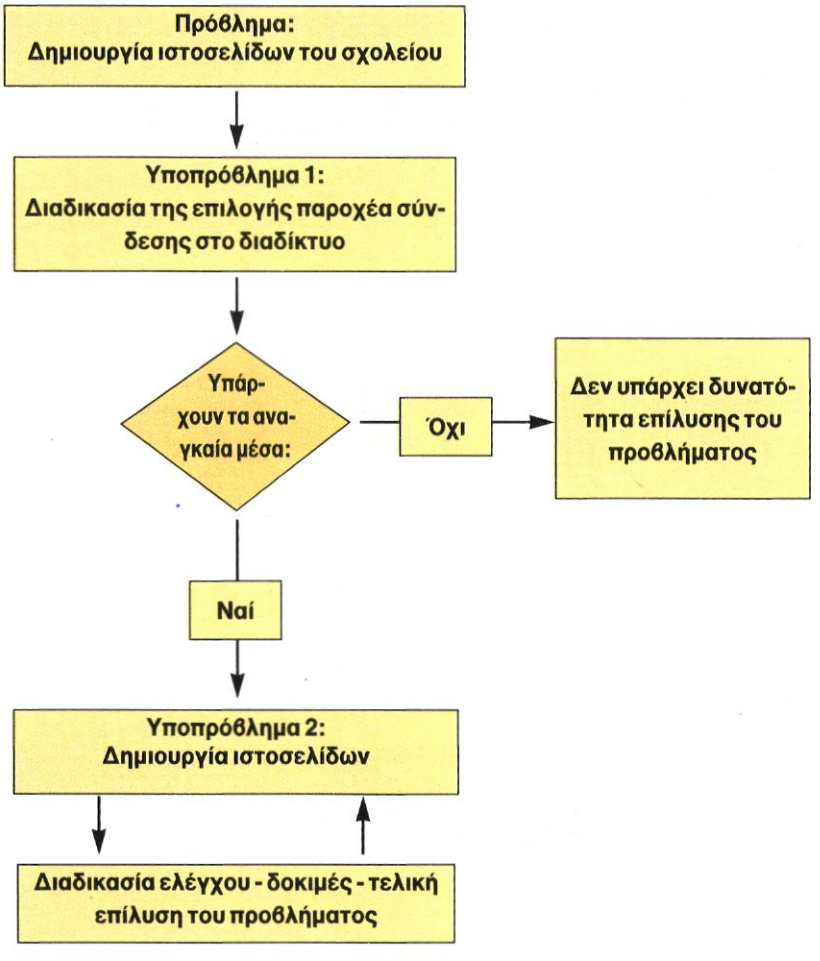
του σχολείου στο διαδίκτυο.
Υποπρόβλημα 2: Η δημιουργία και ο σχεδιασμός των επιθυμητών ιστοσελίδων.

Όσον αφορά στον τρόπο εμφάνισης των σελίδων στο διαδίκτυο, πρέπει να αντιμετωπίσουμε τα εξής ζητήματα:

- Υπάρχουν διαθέσιμα χρήματα για την αναγκαία σύνδεση μέσω κάποιου παροχέα πρόσβασης στο διαδίκτυο καθώς και για το κόστος "ενοικίασης" του χώρου, που θα φιλοξενήσει τις σελίδες μας; Αν η απάντηση στο ερώτημα αυτό είναι αρνητική και ταυτόχρονα δεν υπάρχει εναλλακτική λύση, η οποία να μη απαιτεί κάποια δαπάνη (π.χ.,, δωρεάν παροχής προς το σχολείο), τότε προφανώς δεν μπορούμε να προχωρήσουμε τα σχέδια μας. Αν όμως υπάρχει αυτή η δυνατότητα, τότε περνάμε στο επόμενο ζήτημα
- Αν δεν υπάρχει ήδη παροχέας πρόσβασης στο διαδίκτυο για το σχολείο, πρέπει να επιλέξουμε κάποιον μετά από έρευνα στην αγορά, καθορίζοντας ταυτόχρονα και τις προϋποθέσεις για την επιλογή αυτή (κόστος σύνδεσης και φιλοξενίας σελίδων, ταχύτητες πρόσβασης, πλήθος παρεχόμενων λογαριασμών σύνδεσης κ.λ.π.).
- Με ποια διαδικασία θα γίνεται η ενημέρωση με τυχόν αλλαγές, ή προσθήκες των ιστοσελίδων μας;

Ως προς το δεύτερο υποπρόβλημα, τα βασικά ερωτήματα που πρέπει να απαντηθούν είναι:
- Ποιο θα είναι το περιεχόμενο και ποια θα είναι η εμφάνιση των ιστοσελίδων;
- Με ποιο εργαλείο λογισμικού θα αναπτυχθούν οι ιστοσελίδες, ποιοι θα εργαστούν και ποιο μέρος του όλου έργου θα αναλάβει ο κάθε ένας από τους συμμετέχοντες;

Εφόσον απαντηθούν τα ερωτήματα αυτά, μπορούμε να ξεκινήσουμε την εργασία μας, δημιουργώντας με τα επιλεχθέντα εργαλεία λογισμικού τις ιστοσελίδες που θέλουμε. Η παραπάνω διαδικασία απεικονίζεται διαγραμματικά στο **σχήμα 4.3.**



Πως θα τροποποιούσατε το παράπλευρο σχήμα, εάν θα χρειαζόταν να εξετάσετε και τη δυνατότητα της δωρεάν φιλοξενίας των σελίδων σας από κάποιο παροχέα;

Σχήμα 4.3 . - Διαδικασία επίλυσης του προβλήματος της δημιουργίας ιστοσελίδων ενός σχολείου.

ΠΑΡΑΤΗΡΗΣΗ

Η διαδικασία που ακολουθήθηκε παραπάνω δεν είναι απαραίτητα ο μοναδικός τρόπος αντιμετώπισης του συγκεκριμένου προβλήματος. Είναι δυνατό κάποιος να επιλέξει διαφορετική πορεία. Σε κάθε περίπτωση όμως, τα βήματα που θα σχεδιασθούν πρέπει να τοποθετηθούν με μια λογική σειρά ώστε να επιτρέπουν τον καλλίτερο δυνατό καταμερισμό εργασιών. Επίσης η όλη πορεία θα πρέπει να είναι δυναμική, επιτρέποντας μετά από έλέγχους και δοκιμές στο κάθε βήμα της, την περιορισμένη αναθεώρηση όποιου μέρους κριθεί σκόπιμο, για την καλύτερη εξυπηρέτηση του τελικού στόχου.

*Αν και ο όρος **σύστημα** χρησιμοποιήθηκε εκτεταμένα από το 1950 και μετά, η ιδέα ότι η ολότητα έχει ιδιότητες μη αναγώγιμες στα μέρη της, ήταν γνωστή από την κλασσική αρχαιότητα. "Το γαρ όλον πρότερον του μέρους εστί" Αριστοτέλης, Πολιτικά.*

4.2. ΟΡΙΣΜΟΙ – ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΣΥΣΤΗΜΑΤΩΝ

Ο όρος "σύστημα" χρησιμοποιήθηκε σαν αντικείμενο επιστημονικής έρευνας και αυστηρής αντιμετώπισης κατά τα τελευταία 50 χρόνια και με διαφορετικούς ορισμούς, ανάλογα με την επιστημονική σφαίρα στην οποία ανήκε το εκάστοτε αντικείμενο έρευνας ή το πεδίο εφαρμογής (φιλοσοφία, μαθηματικά, φυσική, μηχανική, βιολογία, κοινωνιολογία κ.λ.π.).

Ορισμός

Ως **σύστημα** νοείται μια συλλογή από αλληλοσυσχετιζόμενα μέρη τα οποία δρουν σα σύνολο για να πετύχουν ένα κοινό σκοπό.

Ειδικότερα, τα συστήματα που θα μας απασχολήσουν στο βιβλίο αυτό είναι συστήματα που έχουν σα σκοπό την επεξεργασία, αποθήκευση, αναζήτηση κ.λ.π. πληροφοριών και τα οποία ονομάζονται πληροφοριακά συστήματα. Παραδείγματα τέτοιων συστημάτων είναι:

- ένα σύστημα πρωτοκόλλησης εγγράφων μιας επιχείρησης ή ενός οργανισμού,
- το σύστημα παρακολούθησης των αποθεμάτων μιας αποθήκης,
- το σύστημα έκδοσης διαβατηρίων σε μια νομαρχία,
- το σύστημα παρακολούθησης των αδειών του προσωπικού μιας εταιρείας,
- το σύστημα οικονομικής διαχείρισης ενός εμπορικού καταστήματος,
- το σύστημα υποστήριξης μιας δανειστικής βιβλιοθήκης,
- το σύστημα τήρησης ιστορικού των νοσηλευθέντων ενός νοσοκομείου,
- το σύστημα προγραμματισμού πτήσεων ενός αεροδρομίου,
- το σύστημα κράτησης θέσεων για ένα ποδοσφαιρικό αγώνα κ.λ.π.

Τα ουσιώδη, σταθερά και αλληλένδετα μέρη που απαρτίζουν το σύστημα λέγονται **στοιχεία** του συστήματος. Π.χ., σα στοιχεία του συστήματος πρωτοκόλλησης εγγράφων, το οποίο αναφέρθηκε παραπάνω, θα μπορούσαν να θεωρηθούν οι υπάλληλοι που εργάζονται στο πρωτόκολλο, το βιβλίο πρωτοκόλλησης, το αρχείο, οι θυρίδες καταχώρισης, τα χρησιμοποιούμενα μέσα κ.λ.π.

Τα συστήματα διακρίνονται σε ανοικτά και κλειστά. **Ανοικτά** ονομάζονται τα συστήματα που επηρεάζονται από το περιβάλλον τους (ή το επηρεάζουν), ενώ **κλειστά** ονομάζονται εκείνα που δεν επηρεάζονται από το περιβάλλον τους, ούτε το επηρεάζουν. Τα πληροφοριακά συστήματα είναι ανοικτά συστήματα, γιατί επεξεργάζονται πληροφορίες που προέρχονται από το περιβάλλον τους, τις δε πληροφορίες που επεξεργάζονται τις διαθέτουν σε αυτό.

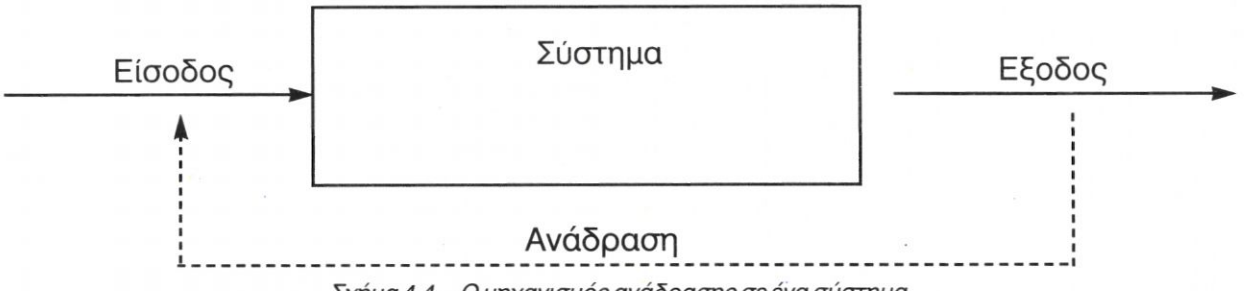
ΠΑΡΑΤΗΡΗΣΗ

Στον πραγματικό κόσμο δεν υπάρχουν κλειστά συστήματα, δεδομένου ότι όλα τα συστήματα δέχονται επιδράσεις από το εξωγενές τους περιβάλλον, έστω και ασθενείς. Σε ορισμένες μόνο περιπτώσεις και κάτω από συγκεκριμένες παραδοχές μπορούμε να θεωρήσουμε κλειστά συστήματα. Π.χ., στο εργαστήριο της φυσικής, προκειμένου να μελετήσουμε τους νόμους των αερίων, θεωρούμε πειραματικές διατάξεις εντελώς απομονωμένες από το περιβάλλον τους. (Π.χ., ο νόμος της διαστολής ισχύει και επαληθεύεται μόνο κάτω από σταθερές συνθήκες πίεσης και θερμοκρασίας).

Ένα σύστημα έχει τα εξής χαρακτηριστικά:

- **Δομή**. Η σχετικά σταθερή οργάνωση του συστήματος, δηλαδή η διάταξη των στοιχείων του στο χώρο και το χρόνο.
- **Όρια**. Η νοητή γραμμή που περιβάλλει τα στοιχεία του συστήματος και τις διασυνδέσεις τους και η οποία το καθιστά (σχετικά) αυτόνομο σύνολο και αναγνωρίσιμο. Ότι είναι μέσα σ' αυτήν ανήκει στο σύστημα, ενώ ό,τι είναι έξω απ' αυτήν, ανήκει στο περιβάλλον του.
- **Περιβάλλον**. Το σύνολο των παραγόντων που επηρεάζουν το σύστημα και επηρεάζονται από αυτό.
- **Είσοδος(οι)** (*input*). Ορισμένη επίδραση του περιβάλλοντος στο σύστημα (ό,τι μπαίνει μέσα στο σύστημα και επηρεάζει τη συμπεριφορά του). Οι είσοδοι συχνά αναφέρονται και σαν δεδομένα εισόδου, εισερχόμενα δεδομένα ή σαν εισροές του συστήματος.
- **Έξοδος(οι)** (*output*). Ορισμένη επίδραση του συστήματος στο περιβάλλον (ό,τι βγαίνει από το σύστημα προς το περιβάλλον). Οι έξοδοι συχνά αναφέρονται και σαν δεδομένα εξόδου, εξερχόμενα δεδομένα ή σαν εκροές του συστήματος. Οι είσοδοι και έξοδοι μπορεί να είναι διάφορες μορφές ύλης, ενέργειας ή πληροφορίας.
- **Μετατροπή**. Η διαδικασία επεξεργασίας και μετασχηματισμού των εισόδων σε εξόδους. Σ' αυτή συμμετέχουν όλα τα στοιχεία του συστήματος, καθένα με τη λειτουργία του.
- **Έλεγχος**. Ο μηχανισμός που διατηρεί τη δομή και λειτουργίες του συστήματος και εξασφαλίζει τις εξόδους του. Στο μηχανισμό αυτό βασική είναι η λειτουργία της **ανάδρασης** (ή ανατροφοδότησης) (*feedback*) **(σχήμα 4.4.)**. Ο μηχανισμός ανάδρασης συγκρίνει τα παρατηρούμενα αποτελέσματα (έξοδοι) με τα επιθυμητά και με το αποτέλεσμα της σύγκρισης (π.χ., την παρατηρούμενη διαφορά) τροφοδοτεί το σύστημα και ειδικότερα τους ρυθμιστικούς μηχανισμούς που ελέγχουν τη λειτουργία του.

Μέσω του μηχανισμού ανάδρασης ένα σύστημα ελέγχει τις εκροές του, τις συγκρίνει με τις επιθυμητές και, σε περίπτωση διαφοράς, προβαίνει σε κατάλληλες ρυθμίσεις προς την κατεύθυνση της επίτευξης των στόχων του.

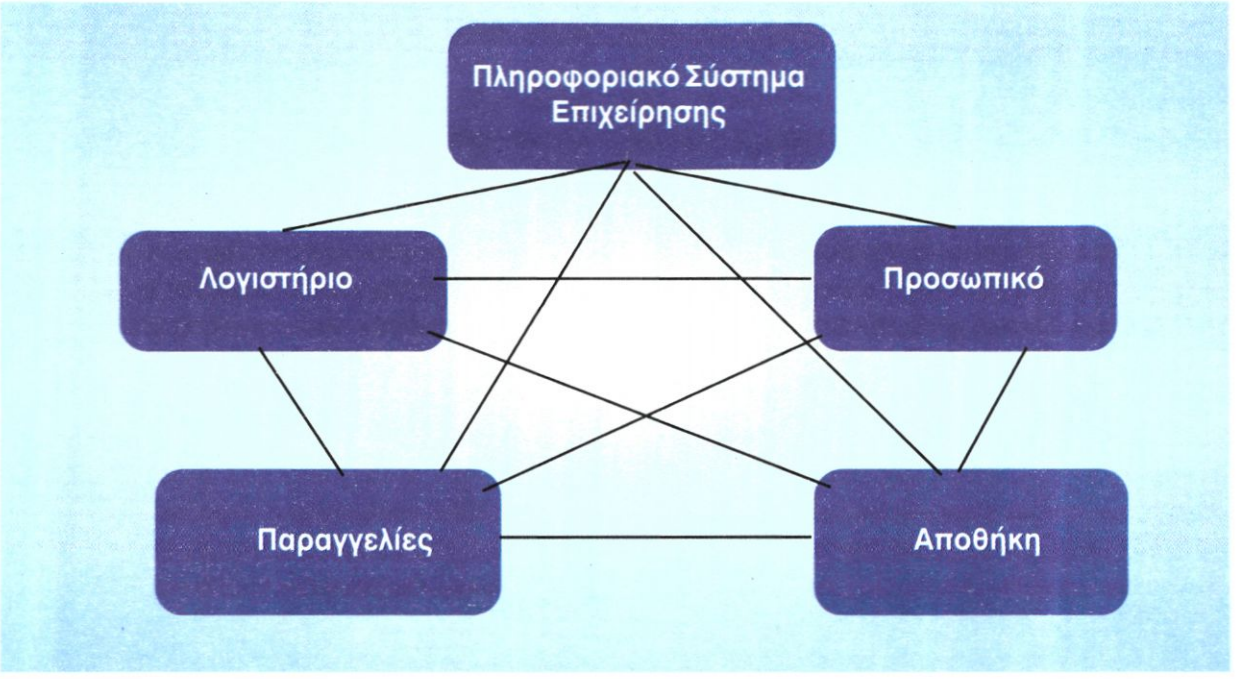


Σχήμα 4.4. - Ο μηχανισμός ανάδρασης σε ένα σύστημα.

Όλα τα συστήματα μπορούν να χωρισθούν σε **υποσυστήματα**, δηλαδή αυτοτελείς λειτουργικές ενότητες που επικοινωνούν μεταξύ τους, επηρεάζοντας έτσι καθένα τη λειτουργία των άλλων.

Ορισμός

Το πιο κάτω **σχήμα 4.5.** απεικονίζει με απλό τρόπο τη δομή και τις διασυνδέσεις μεταξύ των επιμέρους υποσυστημάτων, που συγκροτούν το αυτοματοποιημένο πληροφοριακό σύστημα μιας υποθετικής μικρής επιχείρησης λιανικού εμπορίου. Οι γραμμές που διασυνδέουν τα υποσυστήματα, δείχνουν ότι υπάρχει κάποιου είδους επικοινωνία ή ανταλλαγή στοιχείων μεταξύ του.



Σχήμα 4.5. – Διαγραμματική απεικόνιση ενός τυπικού Συστήματος με τα Υποσυστήματά του.

Όταν μελετάμε ένα πληροφοριακό σύστημα, με σκοπό να εντοπίσουμε οργανωτικά ή λειτουργικά προβλήματα που μπορεί να υπάρχουν, με απώτερο σκοπό να σχεδιάσουμε μια εφαρμογή λογισμικού που θα υποστηρίζει αποτελεσματικά το σύστημα, χρησιμοποιούμε μια μεθοδολογία που ονομάζεται **Ανάλυση Συστημάτων.** Κατ' επέκταση **αναλυτής συστημάτων** είναι εκείνος που χρησιμοποιεί τη μεθοδολογία της ανάλυσης συστημάτων για να προσεγγίσει και να μελετήσει ένα σύστημα για τους σκοπούς που αναφέρθηκαν παραπάνω. Προφανώς της ανάλυσης ακολουθεί η σχεδίαση ενός νέου συστήματος, που ελπίζεται ότι θα είναι καλύτερο από το προηγούμενο.

Πολλά από τα οργανωτικά και λειτουργικά προβλήματα οφείλονται στην αδυναμία του ανθρώπινου νου να αποθηκεύσει και να επεξεργαστεί αποδοτικά μεγάλο όγκο πληροφοριών. Οι εφαρμογές λογισμικού εκμεταλλεύονται τις τεράστιες δυνατότητες των σύγχρονων ψηφιακών υπολογιστών και δίνουν λύση σε πλήθος προβλημάτων που εμφανίζονται σε ένα τυπικό οργανωτικό περιβάλλον.

Στην ανάλυση συστημάτων είναι βασική η έννοια της διαδικασίας. **Διαδικασία** είναι μία ακριβής και λεπτομερής ακολουθία γραπτών ή άγραφων οδηγιών, που βήμα προς βήμα εξηγεί: α) **τι** πρέπει να γίνει, β) **ποιος** θα το κάνει, γ) **πότε** θα γίνει και δ) **πως** θα γίνει. Επί παραδείγματι, στο σύστημα έκδοσης διαβατηρίων που προαναφέρθηκε στα παραδείγματα του κεφαλαίου 4.2, μια χαρακτηριστική διαδικασία είναι η διαδικασία ανανέωσης του διαβατηρίου. Αυτή προσδιορίζει επακριβώς πως γίνεται η ανανέωση ενός διαβατηρίου, πότε, κάτω από ποιες προϋποθέσεις, ποιος είναι αρμόδιος να την κάνει κ.λ.π.

Αν και έχουν προταθεί διάφορες κατηγοριοποιήσεις συστημάτων, ενδιαφέρον για την προσέγγιση που γίνεται στο παρόν βιβλίο παρουσιάζει η διάκρισή τους σε δύο μεγάλες κατηγορίες:

ΚΑΤΗΓΟΡΙΕΣ ΣΥΣΤΗΜΑΤΩΝ

I. Τα φυσικά συστήματα *(natural systems)*

Π.χ.
- Τα αστρικά συστήματα (γαλαξίες, ηλιακό σύστημα).
- Τα γεωλογικά συστήματα (ποταμοί, οροσειρές, δάση).
- Τα μοριακά συστήματα (σύνθετα συμπλέγματα ατόμων) κ.α.

II. Τα συστήματα φτιαγμένα από ανθρώπους *(man-made systems)*

Π.χ.
- Τα κοινωνικά συστήματα (νομοθεσία, κοινωνική ασφάλιση).
- Τα συστήματα μεταφοράς ανθρώπων - οχημάτων (δίκτυα δρόμων, αεροπορικές γραμμές, πλοία).
- Τα διάφορα συστήματα παραγωγής (εργοστάσια, βιοτεχνίες).
- Τα συστήματα επικοινωνίας (τηλεφωνία, τηλεομοιοτυπία).

ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Συστήματα ενταγμένα στη δεύτερη κατηγορία, χρησιμοποιούν στις μέρες μας όλο και περισσότερο τους υπολογιστές, χωρίς όμως κάτι τέτοιο να είναι υποχρεωτικό. Σε πολλές μάλιστα περιπτώσεις, σχετικές μελέτες σκοπιμότητας αποδεικνύουν ότι δεν είναι πάντα σκόπιμο να υιοθετούνται λύσεις που στηρίζονται στη χρήση των υπολογιστών, για ποικίλους λόγους (κόστος, ασφάλεια, ευελιξία, κ.λ.π.). Αυτό βέβαια δεν αποκλείει να γίνει κάτι τέτοιο σε μια επόμενη χρονική στιγμή.

Με βάση τα όσα πιο πάνω αναφέρθηκαν, οδηγούμαστε και σε μια νέα κατηγορία συστημάτων, που στην ουσία αποτελεί υποκατηγορία των "Από ανθρώπους διαχειριζόμενων συστημάτων". Πρόκειται για τα **αυτοματοποιημένα συστήματα**, που ελέγχονται από έναν ή περισσότερους ηλεκτρονικούς υπολογιστές.

Αν και στην πράξη υπάρχουν τόσο διαφορετικά ως προς την φιλοσοφία και το σχεδιασμό τους **αυτοματοποιημένα συστήματα**, όλα απαρτίζονται από κάποιες κοινές συνιστώσες. Κυριότερες από αυτές είναι οι εξής:

- **Υλικό υπολογιστή** *(computer hardware)*. Όλα όσα συνθέτουν τα υλικά μέρη ενός υπολογιστή και των κάθε είδους περιφερειακών τους (κεντρικές μονάδες επεξεργασίας, δίσκοι, τερματικά, εκτυπωτές κ.ο.κ.). Το υλικό υπολογιστή αναφέρεται συχνά και σαν υλικό συστήματος.
- **Λογισμικό υπολογιστή** *(computer software)*. Λειτουργικά συστήματα, προγράμματα διαχείρισης δικτύων, εφαρμογές λογισμικού κ.ο.κ. αποτελούν λογισμικό απαραίτητο για την αξιοποίηση του υπολογιστή. Το λογισμικό υπολογιστή αναφέρεται συχνά και σαν λογισμικό συστήματος.

Μηχανοργάνωση είναι η μετατροπή ενός χειρογραφικού συστήματος οργάνωσης σε αυτοματοποιημένο, με τη χρήση εφαρμογών λογισμικού και κατάλληλου μηχανογραφικού εξοπλισμού

- **Ο ανθρώπινος παράγοντας**. Συντίθεται από όλους όσους χρησιμοποιούν το σύστημα, επικοινωνούν μαζί του και πραγματοποιούν με τη βοήθειά του τις καθημερινές τους εργασίες.
- **Δεδομένα**. Αποτελούνται από όλες τις πληροφορίες των οποίων η διαχείριση γίνεται μέσα από τις εφαρμογές των υπολογιστών.
- **Διαδικασίες**. Όλες οι οργανωμένες ενότητες εντολών, οι οποίες χρησιμοποιούνται από το λογισμικό για την αυτοματοποίηση επιμέρους εργασιών.

ΜΕΓΑΛΕΣ ΚΑΤΗΓΟΡΙΕΣ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

Συστήματα άμεσης επικοινωνίας
(on-line systems)

Στα συστήματα αυτά η είσοδος και η έξοδος πληροφοριών γίνεται άμεσα, έτσι ώστε ο χρήστης να βλέπει απευθείας τα αποτελέσματα των αλλαγών που εισάγει στο σύστημα, τη στιγμή που τις εισάγει και από το περιβάλλον που τις εισάγει. Π.χ., ο υπάλληλος ενός βιβλιοπωλείου από όπου ένας πελάτης αγοράζει ένα βιβλίο, πληκτρολογεί τον κωδικό, βάσει του οποίου αυτό είναι καταχωρημένο στο υποσύστημα της αποθήκης και έτσι ενημερώνει άμεσα το σύστημα ότι τα αποθέματα του συγκεκριμένου τίτλου μειώθηκαν κατά ένα. Ταυτόχρονα εκτυπώνεται απόδειξη λιανικής πώλησης ή τιμολόγιο, ενημερωμένο με τα στοιχεία της αγοράς και δίνεται στον πελάτη για την εξόφληση.

Συστήματα μη άμεσης επικοινωνίας
(off-line systems)

Στα συστήματα αυτά η είσοδος και η έξοδος πληροφοριών δεν γίνεται άμεσα. Ο χρήστης βλέπει το αποτέλεσμα των αλλαγών που εισάγει στο σύστημα σε μεταγενέστερο χρόνο. Π.χ., στο σύστημα έκδοσης διαβατηρίων μιας νομαρχίας, ο υπάλληλος εισάγει από την οθόνη του υπολογιστή τα στοιχεία του πολίτη που ζητά να του εκδοθεί διαβατήριο. Το σύστημα διαβιβάζει τα στοιχεία αυτά στην κεντρική βάση δεδομένων που είναι εγκατεστημένη στο υπουργείο εσωτερικών, προκειμένου να διαπιστωθεί εάν ο πολίτης έχει απαγόρευση εξόδου από τη χώρα, ή αν έχει βγάλει διαβατήριο από άλλη νομαρχία (οπότε δεν δικαιούται διαβατηρίου). Επειδή το σύστημα είναι μη άμεσης επικοινωνίας, η απάντηση από τον κεντρικό υπολογιστή του υπουργείου μπορεί να αποσταλεί στη νομαρχία μετά από μερικά λεπτά, μερικές ώρες ή ακόμη και την επόμενη ημέρα.

Συστήματα πραγματικού χρόνου
(real time systems)

Σαν τέτοια ορίζονται συστήματα, τα οποία ελέγχουν ένα περιβάλλον, που περιλαμβάνει είσοδο-έξοδο στοιχείων, επεξεργασία τους και επιστροφή των αποτελεσμάτων με τέτοιο τρόπο, ώστε ο επηρεασμός του περιβάλλοντος από τις πραγματοποιούμενες αλλαγές να γίνεται "πολύ γρήγορα". Η έννοια του "πολύ γρήγορα" είναι εντελώς σχετική και εξαρτάται από τη φύση του συστήματος. Για παράδειγμα, άλλες είναι οι απαιτήσεις ταχύτητας στα μηχανήματα συναλλαγών των τραπεζών, άλλες στα συστήματα ελέγχου στη βιομηχανία, άλλες στα πυραυλικά συστήματα ελέγχου και άλλες σε περιπτώσεις όπου πρέπει να ενημερωθεί ένα μεγάλο αρχείο μιας επιχείρησης για κάποια αλλαγή, που θα γίνει σε κάποιο πεδίο του και αφορά μερικές χιλιάδες εγγραφές.

Συστήματα υποστήριξης αποφάσεων
(Decision support systems)

Τα συστήματα αυτά διαχειρίζονται τα δεδομένα τους με τρόπο που επιτρέπει σε όποιον τα χρησιμοποιεί, να μπορεί να τα αξιοποιήσει στην κατεύθυνση της λήψης αποφάσεων πάνω σε σημαντικά ζητήματα στρατηγικής ή προγραμματισμού της επιχείρησης ή του οργανισμού όπου εργάζεται. Χαρακτηριστικά παραδείγματα τέτοιων συστημάτων είναι ορισμένα γνωστά και δημοφιλή πακέτα (Lotus, Excel, Framework κ.ο.κ.), αλλά και μερικά αρκετά ισχυρότερα όσο και πολύπλοκα σύγχρονα συστήματα, που στηρίζονται πάνω σε Βάσεις Δεδομένων, όπως είναι τα συστήματα σύγχρονης αναλυτικής επεξεργασίας *(On-line Analytical Processing ή OLAP για συντομία).*

Συστήματα διαχείρισης γνώσης
(Knowledge-based systems)

Είναι συστήματα που στον πυρήνα τους ενσωματώνουν τεχνικές και αλγορίθμους "τεχνητής νοημοσύνης" *(artificial intelligence).* Είναι γνωστά και με τον όρο "έμπειρα συστήματα" *(expert systems).* Τα συστήματα αυτά χρησιμοποιούν μια "βάση γνώσης" *(knowledge base),* δηλαδή μια βάση δεδομένων *(data base),* εφοδιασμένη με ένα μηχανισμό εξαγωγής συμπερασμάτων *(inference engine).*

4.3. ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΣΥΓΚΕΝΤΡΩΣΗΣ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑΣ ΔΕΔΟΜΕΝΩΝ

Ο καθορισμός του προβλήματος, που, όπως αναφέρθηκε στην αρχή του κεφαλαίου, αποτελεί το πρώτο βήμα στη διαδικασία της ανάλυσης, απαιτεί να συγκεντρωθούν και να επεξεργαστούν όλα τα απαραίτητα στοιχεία και πληροφοριακά δεδομένα, που είναι σχετικά με το πρόβλημα. Η συγκέντρωση των δεδομένων αυτών εξαρτάται από το εάν το πρόβλημα είναι η ανάπτυξη (κατασκευή) μιας εφαρμογής λογισμικού γενικής χρήσης, ή μιας εφαρμογής λογισμικού ειδικής χρήσης.

- Οι εφαρμογές **γενικής χρήσης** απευθύνονται σε μεγάλες κατηγορίες χρηστών, οι οποίοι έχουν παρεμφερείς ανάγκες. Τέτοιες εφαρμογές είναι π.χ., οι επεξεργαστές κειμένου, τα προγράμματα επεξεργασίας ηλεκτρονικού λογιστικού φύλλου, τα διάφορα σχεδιαστικά προγράμματα, τα συνήθη συστήματα διαχείρισης βάσεων δεδομένων κ.λ.π. Στην περίπτωση ανάπτυξης μιας εφαρμογής λογισμικού γενικής χρήσης τα δεδομένα της ανάλυσης καθώς και οι γενικοί και ειδικοί στόχοι της εφαρμογής προσδιορίζονται από την ίδια την εταιρεία ανάπτυξης του λογισμικού και έχουν σαν στόχο να καλύψουν τις ανάγκες μιας όσο το δυνατόν μεγαλύτερης κατηγορίας χρηστών, οι οποίοι θα θελήσουν να αγοράσουν την εφαρμογή, **αφού τη δουν ολοκληρωμένη**.

- Οι εφαρμογές **ειδικής χρήσης** σχεδιάζονται και αναπτύσσονται με σκοπό να καλύψουν τις ειδικές ανάγκες ενός συγκεκριμένου χρήστη. Ο χρήστης αυτός μπορεί να είναι ένα ολόκληρο Τμήμα, μια Διεύθυνση, ένας υπάλληλος μιας Οργάνωσης, δηλαδή ενός Υπουργείου, μιας Τράπεζας, μιας ιδιωτικής επιχείρησης κ.λ.π., ή ένας απλός ιδιώτης. Στην περίπτωση ανάπτυξης μιας εφαρμογής λογισμικού ειδικής χρήσης τα δεδομένα της ανάλυσης καθώς και οι γενικοί και ειδικοί στόχοι της εφαρμογής συγκεντρώνονται και προσδιορίζονται μέσα από στενή συνεργασία του αναλυτή συστήματος με τον μελλοντικό της χρήστη, για τον

οποίο και αναπτύσσεται η εφαρμογή και ο οποίος στο εξής θα αναφέρεται και σαν "**πελάτης**". Σε πολλές περιπτώσεις, όταν το πρόβλημα είναι πολύπλοκο, ή όταν ο χρόνος ανάπτυξης μιας εφαρμογής λογισμικού είναι περιορισμένος, την ανάλυση αναλαμβάνει μια ομάδα αναλυτών. Αν και συνήθως ο αναλυτής συστημάτων είναι και προγραμματιστής (δηλαδή κατασκευάζει με τη χρήση μιας γλώσσας προγραμματισμού ηλεκτρονικών υπολογιστών τα προγράμματα που συνθέτουν την εφαρμογή - εξ ου και ο όρος αναλυτής/προγραμματιστής), συχνά συνεργάζεται με άλλους προγραμματιστές, οι οποίοι αναλαμβάνουν την υλοποίηση της εφαρμογής. Όμως ο αναλυτής είναι εκείνος που έχει τον γενικό έλεγχο της ανάπτυξης της εφαρμογής, από τα πρώτα στάδια της ανάλυσης και σχεδίασης, έως την υλοποίηση της εφαρμογής, τον έλεγχο, την παρακολούθηση καλής λειτουργίας της, την επαλήθευση και αξιολόγησή της. Σήμερα δεν υπάρχει ουσιαστική διαφορά μεταξύ αναλυτών και προγραμματιστών εξαιτίας του γεγονότος ότι οι σύγχρονες γλώσσες προγραμματισμού προσδιορίζουν και τους όρους με τους οποίους πρέπει να γίνεται η ανάλυση ενός συστήματος. Συνεπώς ο αναλυτής πρέπει να είναι και προγραμματιστής και το αντίστροφο.

Ανεξάρτητα από το αν μια εργασία ανάλυσης συστήματος γίνεται με σκοπό την ανάπτυξη μιας εφαρμογής γενικής ή ειδικής χρήσης, βασικά στοιχεία που πρέπει εξ αρχής να καθορίσει ο αναλυτής είναι

- το **αντικείμενο** της εφαρμογής (δηλαδή σε ποιο ακριβώς πρόβλημα θα δίνει λύση),
- το **εύρος** της εφαρμογής (δηλαδή ποιες λειτουργίες του συστήματος θα περιλαμβάνει) και
- τους **στόχους** της εφαρμογής (δηλαδή το τι ακριβώς θα πρέπει να κάνει η εφαρμογή).

Τα βασικά εργαλεία που χρησιμοποιεί ο αναλυτής για να συγκεντρώσει τις απαραίτητες πληροφορίες και δεδομένα για τον καθορισμό των παραπάνω είναι τα **ερωτηματολόγια** και οι **συνεντεύξεις**.

4.3.1. Τα ερωτηματολόγια

Παράδειγμα:

Γενικά υπάρχουν τρεις μορφές ερωτηματολογίων: τα προκαθορισμένων απαντήσεων, τα ελεύθερων απαντήσεων και τα μικτά. Παράδειγμα ερώτησης με προκαθορισμένες απαντήσεις είναι το εξής:

Ερώτηση:	Απάντηση:
Πόσα βιβλία έχει η βιβλιοθήκη σας;	☐ Λιγότερα από 50 ☐ Από 50 έως 100 ☐ Περισσότερα από 100

Στα ερωτηματολόγια προκαθορισμένων απαντήσεων πρέπει οι απαντήσεις να καλύπτουν όλες τις δυνατές απαντήσεις. Επίσης διευκολύνουν στη μετέπειτα ανάλυση γιατί μπαίνουν εύκολα σε πίνακες και ταξινομούνται και μπορεί να αποτελέσουν δεδομένα για στατιστική ανάλυση χωρίς περαιτέρω επεξεργασία. Επίσης οι ερωτήσεις που τίθενται πρέπει να δοκιμάζονται σε ένα δείγμα ανθρώπων, για να διαπιστωθεί ότι γίνονται εύκολα κατανοητές. Τέλος, έχουν το πλεονέκτημα ότι μπορούν να απαντηθούν χωρίς την πα-

ρουσία του αναλυτή. Τα ελεύθερων απαντήσεων και τα μικτά ερωτηματολόγια συνήθως δίνονται από τον αναλυτή κατά τη διάρκεια μιας συνέντευξης.

Οι ερωτήσεις που περιλαμβάνει ένα τυπικό ερωτηματολόγιο που υποβάλλεται στο πλαίσιο της ανάλυσης ενός συστήματος, συνήθως περιλαμβάνει ερωτήσεις που αφορούν στη λειτουργία του συστήματος, στις εισόδους (δεδομένα εισόδου, στις εξόδους (δεδομένα εξόδου) και στα χρησιμοποιούμενα μέσα. Ειδικότερα:

Οι ερωτήσεις που αναφέρονται στη **λειτουργία** του συστήματος θα πρέπει:
- Να προσδιορίζουν τι γίνεται, από ποιους, πότε και πώς γίνεται, από πού αρχίζει και γιατί γίνεται.
- Να προσδιορίζουν πόσο χρόνο απαιτεί κάθε τι που γίνεται (μέγιστο, ελάχιστο, μέσο όρο).
- Να εντοπίζουν τους ελέγχους που διασφαλίζουν την ακρίβεια των λειτουργιών.

Μπορείτε να αναφέρετε μερικά δεδομένα εισόδου και μερικά δεδομένα εξόδου ενός συστήματος παρακολούθησης των κοινοχρήστων μιας πολυκατοικίας;

Οι ερωτήσεις που αναφέρονται στις **εισόδους** θα πρέπει να προσδιορίζουν:
- Πότε και πώς λαμβάνονται τα δεδομένα εισόδου.
- Από πού και από ποιόν.
- Ποιος είναι ο όγκος τους.
- Αν χρησιμοποιούνται όπως έχουν ή υφίστανται επεξεργασία και τι είδους.
- Ποιος είναι ο χρόνος ζωής των δεδομένων.
- Ποιοι έλεγχοι εξασφαλίζουν την ακρίβεια και πληρότητα των δεδομένων εισόδου.

Οι ερωτήσεις που αναφέρονται στις **εξόδους** θα πρέπει να προσδιορίζουν:
- Τι πληροφορία μεταδίδουν και προς ποιον.
- Πού, πως και γιατί αποθηκεύονται τα δεδομένα εξόδου.
- Ποιος είναι ο όγκος τους.
- Πως επαληθεύεται η ορθότητά τους, δηλαδή τι έλεγχοι γίνονται.
- Ποιες διαδικασίες ανατροφοδότησης υφίστανται.
- Πως εξασφαλίζεται ότι τα δεδομένα εξόδου είναι πλήρη και έγκυρα και παρέχονται έγκαιρα σε εκείνους που τα χρειάζονται.

Οι ερωτήσεις που αναφέρονται στα **χρησιμοποιούμενα μέσα** θα πρέπει να αποκαλύπτουν:
- Τα χρησιμοποιούμενα συστήματα ηλεκτρονικών υπολογιστών.
- Το χρησιμοποιούμενο λογισμικό.
- Το διαθέσιμο προσωπικό.
- Τους χώρους εγκατάστασης του συστήματος.
- Το ύψος του προϋπολογισμού εγκατάστασης και λειτουργίας του συστήματος.

4.3.2. Οι συνεντεύξεις

Εκτός από τα ερωτηματολόγια και τις συνεντεύξεις ο αναλυτής μπορεί να συγκεντρώσει πληροφορίες και από διάφορες άλλες πηγές, όπως είναι το οργανόγραμμα της οργάνωσης του πελάτη, κείμενα περιγραφής εργασιών, χρησιμοποιούμενα ή εκδιδόμενα από την οργάνωση έντυπα, καταστάσεις, πίνακες κ.λ.π., νόμοι ή κανονισμοί που διέπουν τη λειτουργία της οργάνωσης κ.α.

Η προσωπική ικανότητα να συζητά και να ακούει τα σχόλια και τις παρατηρήσεις των άλλων, είναι ίσως το σημαντικότερο προσόν για έναν αναλυτή, γιατί το πιο σημαντικό εργαλείο συλλογής πληροφοριών είναι οι προσωπικές συνεντεύξεις. Υπάρχουν τρεις βασικοί κανόνες για τη διεξαγωγή μιας επιτυχημένης συνέντευξης:

1. Ηθικές κρίσεις από την πλευρά του αναλυτή έχουν θέση στη συζήτηση μόνον όταν συνάδουν με την διαπιστωμένη ηθική εκείνων που συμμετέχουν στη συνέντευξη. Συνεπώς, θα πρέπει ο αναλυτής να προσπαθεί να προβλέπει το ηθικό υπόβαθρο (σύστημα αξιών) του ερωτώμενου. Και αυτό για δύο λόγους: πρώτον για να μη βρεθεί αντιμέτωπος με εκείνον που ερωτά και δεύτερο για να μπορέσει να εκτιμήσει σωστά το κατά πόσον οι απαντήσεις που παίρνει επηρεάζονται από την ηθική του ερωτώμενου.
2. Ο αναλυτής πρέπει να έχει πάντα κατά νου ότι, στην περίπτωση που ο ερωτώμενος είναι υπάλληλος σε μια Οργάνωση (επιχείρηση, δημόσιο φορέα, κ.λ.π.), ο πρώτος στόχος του ερωτώμενου είναι πάντα (εκτός αν έχει αποδειχθεί το αντίθετο) να διατηρήσει τη θέση που κατέχει στην Οργάνωση, ή να τη βελτιώσει, ή αν αυτό δεν είναι δυνατό, να την αλλάξει με τις μικρότερες δυνατές απώλειες, όπως αυτός (ή αυτή) τις εννοεί. Συνεπώς οι απαντήσεις θα πρέπει να εξετάζονται μέσα από αυτό το πρίσμα.
3. Κάθε εργαζόμενος έχει ορισμένες ιδέες για το τι είναι καλό για την Οργάνωση και τι είναι κακό και αυτές τις ιδέες προσπαθεί να διαδώσει και μέσα από αυτό το φίλτρο περνούν και οι απαντήσεις που δίνει. Όμως δεν είναι σε θέση ο κάθε εργαζόμενος, από την περιορισμένη οπτική του να το κρίνει πάντα με ορθότητα. Πρέπει συνεπώς ο αναλυτής να προσπαθεί να διαπιστώνει την ορθότητα των ιδεών και των αντιλήψεων του ερωτώμενου σχετικά με την Οργάνωση και τους στόχους της.

Εάν σε μια συνέντευξη σας εξυπηρετεί να χρησιμοποιήσετε μαγνητόφωνο, θα πρέπει απαραίτητα να εξασφαλίσετε τη συγκατάθεση του ερωτώμενου.

Οι παραπάνω κανόνες δεν εφαρμόζονται πάντα στην πράξη. Πολλοί αναλυτές υιοθετούν ένα προσωπικό στυλ συνεντεύξεων που τους ταιριάζει. Υπάρχουν διάφορα στυλ συνεντεύξεων:

- Η "τυπική" συνέντευξη, δηλαδή η συνέντευξη που ακολουθεί βήμα προς βήμα ένα λογικό σχέδιο, χωρίς περιττές συζητήσεις και οικειότητες, που θα δημιουργούσαν ίσως πιο χαλαρό κλίμα. Αναφέρονται μόνο τα απαραίτητα.
- Η "κανονική" συνέντευξη, δηλαδή η συνέντευξη στην οποία ο ερωτών ακολουθεί κατά γράμμα τους κανόνες της συνέντευξης (βλ. παρακάτω).
- Η "διπλωματική" συνέντευξη, που ο ερωτών προσπαθεί να γίνει συμπαθής στους ερωτώμενους, κάνοντας πολλές φορές διάφορες εξυπηρετήσεις.
- Η "άτυπη" συνέντευξη. Ο ερωτών έχει ένα σχέδιο ερωτήσεων, μπορεί όμως να αποκλίνει από αυτό, χωρίς όμως να χάνει τον έλεγχο της κατάστασης. Το είδος αυτό θέλει προσοχή, γιατί μπορεί ο ερωτώμενος να νομίσει ότι ο αναλυτής έχει έρθει όχι καλά οργανωμένος και προετοιμασμένος για τη συνέντευξη και να μη δώσει την πρέπουσα σημασία στη συζήτηση.

Καλύτερη θεωρείται η συνέντευξη που μπορεί να συνδυάζει και τα τέσσερα παραπάνω στυλ σε κατάλληλες δόσεις. Να μπορεί ο αναλυτής να γίνεται "τυπικός" όταν χρειάζεται, "κανονικός" όταν προκαλείται, "διπλωματικός", όταν υπάρχουν δυσκολίες και "άτυπος", όταν κινείται σε άγνωστο έδαφος.

Όσον αφορά στην οργάνωση μιας συνέντευξης, υπάρχουν γενικά δύο κατηγορίες: η κατευθυνόμενη συνέντευξη και η μη κατευθυνόμενη. Η πρώτη εστιάζεται πάνω στα πρώτα συμπεράσματα που έχει βγάλει ο αναλυτής από την προκαταρκτική μελέτη του συστήματος και μέσω της συνέντευξης προσπαθεί να τα επαληθεύσει ή να τα διαψεύσει. Η δεύτερη έχει σαν στόχο την συγκέντρωση εμπειριών και γνωμών σχετικά με το πρόβλημα. Ανεξάρτητα κατηγορίας, υπάρχουν ορισμένα μεθοδολογικά βήματα - κανόνες, που πρέπει να ακολουθούνται:

- Πρέπει να υπάρχει ένα σχέδιο. Αυτό μπορεί να είναι ένα περίγραμμα της δομής της συζήτησης, ή μπορεί να αποτελείται από κάποιες βασικές ερωτήσεις που πρέπει να τεθούν. Επίσης θα πρέπει, πριν από κάθε συνέντευξη, να έχει προσδιοριστεί ο σκοπός για τον οποίο γίνεται και να έχει εξηγηθεί το γιατί χρειάζεται η γνώμη του ερωτώμενου.
- Πρέπει πάντα να προηγείται ραντεβού. Θα πρέπει η συνέντευξη να γίνεται σε χρόνο που ο ερωτώμενος να μπορεί να αφιερώσει όλη του την προσοχή στη συζήτηση και να μη διακόπτεται από άλλα πράγματα.
- Όσον αφορά στη σειρά των συνεντεύξεων, είναι προτιμότερη η εξής: Πρώτα τα ανώτερα στελέχη της Οργάνωσης, μετά τα μεσαία και τέλος οι απλοί υπάλληλοι. Αυτό επιτρέπει οι ερωτήσεις που θα τεθούν στα χαμηλότερα επίπεδα να τοποθετούνται μέσα στο πλαίσιο των γενικών στόχων της οργάνωσης. Επιπλέον, έτσι εξασφαλίζεται η συγκατάθεση των προϊσταμένων για τη διεξαγωγή των συνεντεύξεων με τους υπαλλήλους τους. Επίσης είναι βασικό να ζητά ο αναλυτής "εξουσιοδοτικό" έγγραφο από τα ανώτερα στελέχη ή τη διαμεσολάβηση προς τα χαμηλότερα, για τη συνδρομή τους στη διεξαγωγή των συνεντεύξεων.
- Καλό είναι ο αναλυτής να συγκεντρώνει προκαταρκτικές πληροφορίες τόσο για τα άτομα που θα πάρει συνέντευξη, όσο και για τα προβλήματα που θα συζητηθούν. Πως εμπλέκεται ο συγκεκριμένος υπάλληλος στη μελετώμενη διαδικασία;
- Δύο ή τρεις σύντομες συνεντεύξεις είναι προτιμότερες από μία μεγάλη, γιατί οι άνθρωποι συχνά κουράζονται και χάνουν το ενδιαφέρον τους σε μαραθώνιες συνεντεύξεις. Επίσης οι πολλές μικρές συνεντεύξεις δίνουν το χρόνο και στον αναλυτή και στον ερωτώμενο να σκεφθεί καλλίτερα και ίσως να επανεκτιμήσει τις απαντήσεις.
- Η γραπτή τεκμηρίωση των συνεντεύξεων είναι ένα άλλο σημαντικό ζήτημα, που πρέπει να προσέχει ο αναλυτής. Πρέπει να κρατά σημειώσεις και σε ορισμένες περιπτώσεις, αν δεν υπάρχουν αντιρρήσεις, να χρησιμοποιεί κασετόφωνο. Ποτέ δεν χρησιμοποιεί κασετόφωνο αν δεν έχει ζητήσει προηγουμένως άδεια από τον ερωτώμενο.
- Στο τέλος της συνέντευξης πρέπει να γίνεται ανακεφαλαίωση, κατά την οποία τονίζονται και επιβεβαιώνονται τα σημεία που υπήρξε συμφωνία με τον ερωτώμενο.

4.4. ΕΚΤΙΜΗΣΗ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ

Απαιτήσεις στην ανάλυση συστημάτων είναι κατά κανόνα οι έξοδοι (εξερχόμενα δεδομένα) μιας εφαρμογής (δηλαδή το τι πρέπει να παράγει η εφαρμογή) και οι διαδικασίες τις οποίες θα υποστηρίζει (δηλαδή το πώς πρέπει να παράγει τις εξόδους). Τυπικές έξοδοι μιας εφαρμογής είναι οι επεξεργασμένες πληροφορίες που παρέχονται σε διάφορες μορφές όπως π.χ., έντυπες καταστάσεις, αναφορές, απαντήσεις σε ερωτήματα που θέτει ο χρήστης, πίνακες στοιχείων κ.λ.π. Μια πρώτη προσέγγιση των απαιτήσεων αυτών γίνεται μέσω των ερωτηματολογίων και των συνεντεύξεων που αναφέρθηκαν παραπάνω. Οι απαιτήσεις αυτές γίνονται απόλυτα συγκεκριμένες στο επόμενο στάδιο δηλ. το στάδιο της ανάλυσης. Πέραν όμως των απαιτήσεων εκροών, υπάρχουν και άλλες, εξίσου σημαντικές για την αποτελεσματική λειτουργία της

Η εισαγωγή δεδομένων παρελθόντων ετών συνήθως γίνεται με πληκτρολόγηση (data entry) ή με σάρωση και ψηφιοποίηση εικόνων εγγράφων (scanning). Στην τελευταία περίπτωση συχνά χρησιμοποιούνται και προγράμματα οπτικής αναγνώρισης χαρακτήρων *(OCR)*, τα οποία μετατρέπουν τις εικόνες εγγράφων σε επεξεργάσιμα κείμενα.

εφαρμογής, οι οποίες θα πρέπει να εκτιμηθούν από τον αναλυτή, σε συνεργασία πάντα με τον "πελάτη". Οι βασικότερες από τις απαιτήσεις αυτές είναι οι εξής:

- **Εισαγωγή δεδομένων παρελθόντων ετών.** Η μετάπτωση ενός χειρογραφικού συστήματος σε αυτοματοποιημένο, μέσω της ανάπτυξης και χρήσης μιας εφαρμογής λογισμικού, συχνά απαιτεί να εισαχθούν στο νέο σύστημα όλα τα πληροφοριακά δεδομένα, που έως τώρα φυλάσσονταν σε χειρόγραφα ή έντυπα αρχεία. Απαιτείται δηλαδή η μετατροπή των χειρογραφικών αρχείων σε αντίστοιχα ηλεκτρονικά, που θα φυλάσσονται σε μαγνητικά μέσα. Π.χ., σε μια εφαρμογή πρωτοκόλλου απαιτείται, πριν από την τελική εγκατάστασή της σε πραγματική λειτουργία, να εισαχθούν όλα τα έγγραφα που πρωτοκολλήθηκαν τα τελευταία χρόνια, ώστε να είναι δυνατή η συσχέτιση νέων εγγράφων με παλαιότερα. Σε μία εφαρμογή μισθοδοσίας, πριν αυτή λειτουργήσει κανονικά, θα πρέπει να εισαχθούν τα ατομικά στοιχεία όλων των υπαλλήλων που έχουν προσληφθεί στο παρελθόν και εξακολουθούν να εργάζονται έως τώρα. Σε μια εφαρμογή δανειστικής βιβλιοθήκης θα πρέπει να εισαχθούν, πριν από την κανονική της λειτουργία, όλα τα διαθέσιμα βιβλία κ.ο.κ. Στον προσδιορισμό της απαίτησης για προκαταρκτική εισαγωγή δεδομένων περιλαμβάνεται, εκτός από το ποια θα είναι αυτά τα δεδομένα και το ποιος θα τα εισάγει (δηλαδή θα τα πληκτρολογήσει) στο σύστημα.
- **Επικοινωνία με άλλα συστήματα.** Συχνά, πολλές από τις πληροφορίες που χρειάζεται μια εφαρμογή για να παράγει τις απαιτούμενες εξόδους, υπάρχουν αποθηκευμένες στα αρχεία άλλων εφαρμογών που λειτουργούν ή πρόκειται να λειτουργήσουν στο χώρο του "πελάτη" ή και εκτός αυτού. Σε αυτές τις περιπτώσεις είναι σκόπιμο να μπορεί η εφαρμογή να επικοινωνεί με τις άλλες εφαρμογές και να παίρνει τις έτοιμες πληροφορίες που χρειάζεται. Το ίδιο συμβαίνει και όταν η εφαρμογή παράγει πληροφορίες που είναι χρήσιμες σε άλλες εφαρμογές που λειτουργούν στο χώρο του "πελάτη". Π.χ., μια εφαρμογή οικονομικής διαχείρισης και προγραμματισμού προμηθειών μπορεί να χρειάζεται ορισμένες από τις πληροφορίες που αποθηκεύει στα αρχεία της μια εφαρμογή παρακολούθησης αποθήκης, που από διετίας λειτουργεί στο χώρο του "πελάτη". Όταν υπάρχουν απαιτήσεις για επικοινωνία μιας εφαρμογής με άλλες, θα πρέπει να προσδιορίζεται με ακρίβεια και σαφήνεια τόσον ο τρόπος της επικοινωνίας (μέσω τοπικού δικτύου, μέσω του διαδικτύου, με μεταφορά των δεδομένων σε δισκέτες κ.λ.π.), όσο και οι πληροφορίες που θα ανταλλάσσονται.
- **Επεκτασιμότητα.** Πολλές φορές ο πελάτης γνωρίζει ή εκτιμά ότι στο άμεσο μέλλον πρόκειται να υπάρξουν νέες ανάγκες, οι οποίες θα πρέπει να καλυφθούν με επέκταση του σχεδιαζόμενου συστήματος. Π.χ., η δημιουργία νέου υποκαταστήματος μιας τράπεζας, η επέκταση του κύκλου εργασιών μιας επιχείρησης σε νέες δραστηριότητες κ.λ.π. Σε αυτή την περίπτωση θα πρέπει να υπάρξει πρόβλεψη ώστε οι μελλοντικές επεκτάσεις του συστήματος να γίνουν εύκολα και με το μικρότερο δυνατό κόστος (π.χ., χωρίς να χρειαστεί να αντικατασταθεί ο μηχανογραφικός εξοπλισμός που θα βρίσκεται ήδη σε λειτουργία). Η πρόβλεψη αυτή συνιστά την απαίτηση ώστε το σχεδιαζόμενο σύστημα να είναι "επεκτάσιμο".
- **Συμβατότητα.** Αν στο χώρο του "πελάτη" υπάρχει ήδη εγκατεστημένος μηχανογραφικός εξοπλισμός και δεν κρίνεται σκόπιμη, η δεν υπάρχει δυνατότητα αντικατάστασής του, τότε θα πρέπει η εφαρμογή να κατασκευαστεί κατά τρόπο ώστε να είναι συμβατή με το συγκεκριμένο υλικό. Επίσης η συμβατότητα θα πρέπει να εξασφαλίζεται και σε επίπεδο λειτουργικού

συστήματος. Με άλλα λόγια η εφαρμογή θα πρέπει να "τρέχει" στο υπάρχον λειτουργικό σύστημα ή στο λειτουργικό σύστημα που επιλέγει ο πελάτης. Επίσης θα πρέπει να είναι συμβατή και με το τυχόν εγκατεστημένο λογισμικό δικτύου, στην περίπτωση που απαιτείται να επικοινωνεί με άλλες εφαρμογές.

Ανάλογα με τη φύση μιας υπό ανάπτυξη εφαρμογής είναι δυνατό να υπάρχουν και άλλοι παράγοντες, που θα πρέπει να εξεταστούν, σε σχέση με τον προσδιορισμό των απαιτήσεων. Κυριότεροι από αυτού είναι οι εξής:

- Αριθμός χρηστών και γεωγραφική τους διασπορά. Σημαντικός παράγοντας από τον οποίο εξαρτάται η επιλογή του μοντέλου ανάπτυξης μιας εφαρμογής είναι ο αριθμός των χρηστών της και η γεωγραφική τους κατανομή. Άλλη αρχιτεκτονική έχει μια εφαρμογή που απευθύνεται σε ένα μεμονωμένο χρήστη, άλλη μια εφαρμογή που θα χρησιμοποιείται ταυτόχρονα από λίγους χρήστες που βρίσκονται στο ίδιο κτίριο και άλλη είναι η αρχιτεκτονική μιας εφαρμογής που θα χρησιμοποιείται από χρήστες διασκορπισμένους σε διαφορετικές πόλεις.
- Χρόνοι απόκρισης του συστήματος. Μια άλλη σημαντική απαίτηση είναι ο μέγιστος αποδεκτός χρόνος απόκρισης μιας εφαρμογής στα διάφορα αιτήματα του χρήστη. Π.χ., σε μια εφαρμογή παρακολούθησης της κίνησης των λογαριασμών καταθέσεων των πελατών μιας τράπεζας, τα τρία δευτερόλεπτα μπορεί να είναι αποδεκτός χρόνος για την εμφάνιση του υπόλοιπου των καταθέσεων ενός πελάτη στην οθόνη του χρήστη, ενώ να είναι απαράδεκτος για την εκτέλεση μιας νέας εγγραφής. Ο χρόνος απόκρισης δεν εξαρτάται μόνον από την αρχιτεκτονική, την τεχνολογία ανάπτυξης και τον τρόπο υλοποίησης της εφαρμογής, αλλά και από τον χρησιμοποιούμενο εξοπλισμό.
- Όγκος πληροφοριακών δεδομένων. Η ορθή εκτίμηση του όγκου των στοιχείων που θα εισάγονται, θα επεξεργάζονται και θα αποθηκεύονται από το σύστημα αποτελεί σημαντικό παράγοντα επιτυχίας στην υλοποίηση μιας εφαρμογής, γιατί προσδιορίζει το μέγεθος των αρχείων και υποδεικνύει το σωστό τρόπο οργάνωσής τους. Μια εφαρμογή μπορεί να λειτουργεί καλά με μικρούς όγκους δεδομένων, αλλά να έχει απαράδεκτους χρόνους απόκρισης, όταν ο όγκος των αρχείων που χρησιμοποιεί μεγαλώσει σημαντικά.
- Το ποσοστό διαθεσιμότητας. Έχει σχέση με την κρισιμότητα της εφαρμογής. Μια σημαντική για την ομαλή λειτουργία της Οργάνωσης του "πελάτη" εφαρμογή πρέπει να είναι πάντοτε διαθέσιμη στους χρήστες της, δηλαδή να μη παρουσιάζει βλάβες ή αστοχίες κατά τη λειτουργία της. Με άλλα λόγια θα πρέπει να έχει ποσοστό διαθεσιμότητας 100%. Λιγότερο σημαντικές εφαρμογές μπορεί να έχουν μικρότερο ποσοστό διαθεσιμότητας. Το ποσοστό διαθεσιμότητας εξαρτάται από πολλούς παράγοντες. Μεταξύ αυτών είναι η αξιοπιστία του εξοπλισμού, οι πραγματοποίηση εκτεταμένων ελέγχων καλής λειτουργίας του λογισμικού κάτω από οριακές συνθήκες, η αρτιότητα εκπαίδευσης των χρηστών, η ύπαρξη εφεδρικού μηχανισμού ασφαλείας σε περιπτώσεις βλαβών κ.α.
- Χρόνος υλοποίησης. Το χρονικό διάστημα μέσα στο οποίο πρέπει να ολοκληρωθεί η ανάλυση, η σχεδίαση, η υλοποίηση, εγκατάσταση και η έναρξη λειτουργίας μιας εφαρμογής, είναι απαίτηση καθοριστική για μια σειρά παραγόντων, όπως είναι το κόστος ανάπτυξης, η μεθοδολογία προσέγγισης, η τεχνολογία υλοποίησης, η ποιότητα του προϊόντος κ.λ.π.
- Εκπαίδευση χρηστών. Ένα νέο μηχανογραφικό σύστημα δεν μπορεί να τεθεί σε παραγωγική λειτουργία, εάν προηγουμένως δεν έχουν εξοικειωθεί με τη χρήση και τη λειτουργία του οι χρήστες. Αυτό γίνεται μέσω κατάλληλων εκπαιδευτικών διαδικασιών (σεμινάρια, εκπαίδευση στις θέσεις εργασίας *(on-the-job training)* κ.λ.π., οι οποίες είναι αναγκαίο να προσδιοριστούν εξ αρχής και να αποτελέσουν δεσμευτική για τον ανάδοχο του έργου απαίτηση.

Ανακεφαλαίωση / περίληψη

Οι εφαρμογές λογισμικού είναι αυτοματοποιημένα συστήματα, τα οποία κατασκευάζονται με σκοπό να δώσουν λύση σε κάποια προβλήματα. Πολλά από τα προβλήματα αυτά οφείλονται στην αδυναμία του ανθρώπινου νου να αποθηκεύσει και να επεξεργαστεί αποδοτικά μεγάλο όγκο πληροφοριών. Οι εφαρμογές λογισμικού εκμεταλλεύονται τις τεράστιες δυνατότητες των σύγχρονων ψηφιακών υπολογιστών και δίνουν λύση σε πλήθος προβλημάτων που εμφανίζονται σε ένα τυπικό οργανωτικό περιβάλλον. Για την ανάπτυξη "ποιοτικών" εφαρμογών λογισμικού, δηλαδή εφαρμογών που ανταποκρίνονται στις ανάγκες και δίνουν λύση στα προβλήματα των χρηστών, έχουν αναπτυχθεί επιστημονικές μέθοδοι και τεχνικές, που ξεκινούν από τον ορθό καθορισμό, ενός προβλήματος και καταλήγουν στη σχεδίαση, υλοποίηση και εγκατάσταση σε πραγματική λειτουργία των εφαρμογών. Η ανάλυση ενός προβλήματος σε υποπροβλήματα είναι μια συνηθισμένη προσέγγιση πολύπλοκων προβλημάτων και αποσκοπεί στη σχεδίαση επιμέρους λύσεων. Στη συνέχεια, από τη σύνθεση των επιμέρους λύσεων προκύπτει η τελική εφαρμογή. Τα ερωτηματολόγια και οι συνεντεύξεις είναι συνηθισμένα εργαλεία που χρησιμοποιούν οι αναλυτές συστημάτων για να συγκεντρώνουν τις απαραίτητες πληροφορίες προκειμένου να προσδιορίσουν το αντικείμενο, το εύρος και τους στόχους μιας εφαρμογής. Επιπλέον χρησιμοποιούνται για τον προσδιορισμό των απαιτήσεων της εφαρμογής, που, εκτός από τις εξόδους της (δηλ. τι πληροφορίες παράγει και σε ποια μορφή), περιλαμβάνουν και άλλες παραμέτρους, όπως είναι ο προσδιορισμός του αριθμού των χρηστών και η γεωγραφική τους διασπορά, ο όγκος των δεδομένων, η ανάγκη για επικοινωνία με άλλα συστήματα, η ανάγκη εισαγωγής στο αυτοματοποιημένο σύστημα δεδομένων παρελθόντων ετών, οι απαιτήσεις συμβατότητας, οι μέγιστοι αποδεκτοί χρόνοι απόκρισης του συστήματος, το ποσοστό διαθεσιμότητας του συστήματος κ.α.

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Γιατί δεν αντιλαμβάνονται όλοι οι άνθρωποι με τον ίδιο τρόπο ένα πρόβλημα;
2. Τι πρέπει να προσδιοριστεί, πριν αρχίσει η προσπάθεια επίλυσης ενός προβλήματος και γιατί;
3. Να θέσετε διάφορα γνωστά σας προβλήματα (από την καθημερινή ζωή) και να προσδιορίσετε συγκεκριμένα βήματα επίλυσής τους.
5. Να ενώσετε με γραμμές τα παρακάτω συστήματα με τις κατηγορίες, στις οποίες ανήκουν.

Σύστημα	Κατηγορία
Το ανθρώπινο σώμα ..... •	• Αυτοματοποιημένα
Το διαδίκτυο ..... •	• Κοινωνικά
Το αυτοκίνητο ..... •	• Φυσικά
Το κράτος ..... •	• Άμεσης Επικοινωνίας *(On-line)*
Το σύστημα κρατήσεων θέσεων της "Ολυμπιακής" . •	• Υποστήριξης αποφάσεων
Το σύστημα πρόγνωσης του καιρού ..... •	• Τεχνολογικά
	• Επικοινωνιακά
	• Διαχείρισης γνώσης

5. Ποια είναι το σημαντικότερα εργαλεία άντλησης πληροφοριών για τον αναλυτή;

6. Αναφέρατε τέσσερις κατηγορίες συνεντεύξεων.
7. Γιατί είναι προτιμότερες δύο ή τρεις μικρές συνεντεύξεις από μία μεγάλη;
8. Αναφέρετε τους δύο γνωστότερους τύπους ερωτηματολογίων.
9. Αναφέρετε τουλάχιστον τρεις σημαντικές απαιτήσεις που πρέπει να προσδιορίζονται πριν από το στάδιο σχεδιασμού και ανάπτυξης μιας εφαρμογής.
10. Να προσδιορίσετε μερικές βασικές απαιτήσεις για μια εφαρμογή λογισμικού που θα αγόραζε το σχολείο, προκειμένου να αυτοματοποιήσει το σύστημα διαχείρισης της δανειστικής του βιβλιοθήκης.

Γλωσσάριο - Ευρετήριο όρων

ανάλυση συστήματος ή συστημάτων *(system ή systems analysis)* - Επιστημονική μέθοδος μελέτης του ρόλου ενός προτεινόμενου συστήματος με κύριο σκοπό τον προσδιορισμό του συνόλου των απαιτήσεων, τις οποίες πρέπει να ικανοποιεί [78].
διαδίκτυο *(internet)* - Το παγκόσμιο δίκτυο των δικτύων [71]
διαδικασία *(procedure)* - Ακριβής και λεπτομερής ακολουθία γραπτών ή άγραφων οδηγιών, που βήμα προς βήμα εξηγεί τι πρέπει να γίνει, ποιος θα το κάνει, πότε θα γίνει και πως θα γίνει [74]
είσοδος(οι) *(input)* - Ορισμένη επίδραση του περιβάλλοντος στο σύστημα (ότι μπαίνει μέσα στο σύστημα και επηρεάζει τη συμπεριφορά του. Συχνά οι είσοδοι αναφέρονται και σαν εισερχόμενα δεδομένα, στοιχεία εισόδου ή εισροές [77]
έξοδος(οι) *(output)* - Ορισμένη επίδραση του συστήματος στο περιβάλλον - δηλαδή ορισμένη συμπεριφορά του συστήματος που επηρεάζει το περιβάλλον. Συχνά οι έξοδοι αναφέρονται και σαν εξερχόμενα δεδομένα, στοιχεία εξόδου ή εκροές [77]
ιστοσελίδες *(web pages)* - Σελίδες υπερκειμένου που δημοσιεύονται στο διαδίκτυο [73]
σύστημα *(system)* - Μια συλλογή από αλληλοσχετιζόμενα μέρη, τα οποία δρουν σαν σύνολο για να επιτύχουν ένα κοινό σκοπό [76]

Βιβλιογραφία

FitzGerald J., FitzGerald A., "Fundamentals of Systems Analysis", John Willey & Sons, 1987.
Γιακουμάκης Εμμ., "Τεχνολογία Λογισμικού – Τόμος Α'", Εκδ. Α. Σταμούλης, 1994.
Χάλαρης Ι., Διαμαντόπουλος Ι., Δουμάς Π., "Δομημένη Ανάλυση Συστημάτων: Από τη Θεωρία στην Πράξη", Εκδ. Νέων Τεχνολογιών, 1992.

ΚΕΦΑΛΑΙΟ 5

ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ ΣΥΣΤΗΜΑΤΟΣ

ΠΕΡΙΕΧΟΜΕΝΑ

5.1. Ανάλυση προβλημάτων – γενικά βήματα
5.2. Προσδιορισμός των απαιτήσεων, που προκύπτουν από την ανάλυση του βασικού προβλήματος σε επιμέρους προβλήματα

ΕΙΣΑΓΩΓΗ

- Θα μπορούσατε να λύσετε ένα μαθηματικό πρόβλημα, αν δεν είχατε κατανοήσει πλήρως τι ζητάει;
- Θα μπορούσατε να απαντήσετε σε μια ερώτηση, αν προηγουμένως δεν την έχετε καταλάβει;
- Θα μπορούσατε να αναλάβετε κάποιο ρόλο ή μια υποχρέωση, αν δεν σας έχουν προηγουμένως διευκρινίσει τις απαιτήσεις αυτού του ρόλου ή της υποχρέωσης;

Ασφαλώς όχι. Κάτι ανάλογο ισχύει και στην περίπτωση ανάπτυξης μιας εφαρμογής λογισμικού.

Ένα από τα σημαντικότερα στάδια (αν όχι το σημαντικότερο) στην ανάπτυξη μιας εφαρμογής λογισμικού (και ενός πληροφοριακού συστήματος, γενικότερα), είναι η ανάλυση απαιτήσεων. Αν η ομάδα που αναπτύσσει το σύστημα δεν έχει αντιληφθεί ακριβώς ποιες είναι οι απαιτήσεις των υποψήφιων χρηστών, είναι αδύνατο να αναπτύξει και να παραδώσει ένα αξιόπιστο και με αντοχή στο χρόνο σύστημα.

Η ομάδα ανάλυσης οφείλει να διαμορφώσει ένα πολύ συγκεκριμένο πλαίσιο επικοινωνίας με τους χρήστες, να καταγράψει τις απαιτήσεις τους και να τους ενημερώνει, ζητώντας την άποψή τους για κάθε εξέλιξη που προκύπτει στην πορεία των εργασιών. Προκειμένου η επικοινωνία αυτή να είναι παραγωγική και δεσμευτική τόσο για τους αναλυτές, όσο και για τους χρήστες, πρέπει να αποτυπώνεται σε φύλλα συνεντεύξεων και ερωτηματολόγια καταγραφής απαιτήσεων, τα οποία συντάσσονται από την ομάδα ανάλυσης και συμπληρώνονται από τους χρήστες.

Για τη σύνταξη της ανάλυσης χρησιμοποιούνται ευρύτατα ειδικά εργαλεία απεικόνισης των κάθε είδους προτύπων, διαδικασιών και λειτουργιών του συστήματος, όπως είναι τα διαγράμματα ροής δεδομένων, τα διαγράμματα HIPO, οι πίνακες αποφάσεων το λεξικό δεδομένων κ.α.

Τελικός στόχος είναι, με το σωστό συνδυασμό όλων των τεχνικών και των εργαλείων που μπορούν να χρησιμοποιηθούν, να προκύψει μια περιεκτική, ακριβής και πλήρης ανάλυση για το νέο σύστημα, αποδεκτή από τους χρήστες του και αξιοποιήσιμη από εκείνους που θα αναλάβουν τον τελικό σχεδιασμό και την σύνταξη των προγραμμάτων.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ - ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να κάνει χρήση τεχνικών για τη συγκέντρωση και επεξεργασία των πληροφοριών που χρειάζεται για να κατανοήσει το πραγματικό πρόβλημα.
2. Να χρησιμοποιεί δομημένα εργαλεία για να περιγράφει και να παρουσιάζει σε τρίτους το πρόβλημα με τρόπο κατανοητό.
3. Να προσδιορίζει με ακρίβεια και σαφήνεια τις απαιτήσεις, με βάση τις οποίες θα αναλυθεί και στη συνέχεια θα σχεδιαστεί η λύση του προβλήματος.

5.1. ΑΝΑΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ – ΓΕΝΙΚΑ ΒΗΜΑΤΑ

Τα πορίσματα του πρώτου σταδίου (βήματος) στη διαδικασία ανάλυσης ενός συστήματος, που περιγράφηκε στο προηγούμενο κεφάλαιο, συνθέτουν τον λεγόμενο **φάκελο της προμελέτης του συστήματος**. Αφού ολοκληρωθεί και εγκριθεί η προμελέτη, το επόμενο στάδιο στην πορεία προς την ανάπτυξη του συστήματος είναι η αναλυτική διερεύνηση και καταγραφή των απαιτήσεων των χρηστών από το νέο σύστημα, ή, όπως είναι ευρύτερα γνωστή, η ανάλυση του συστήματος.

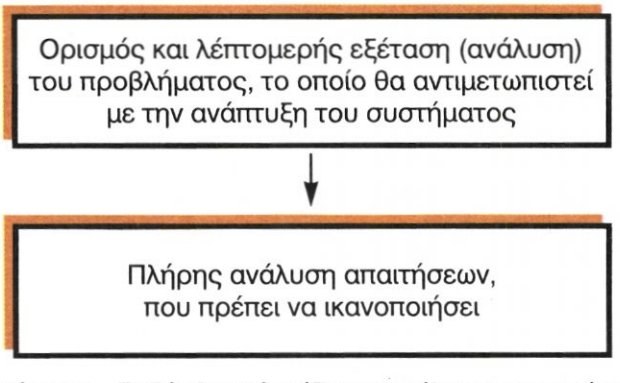
Η πορεία και οι τεχνικές της ανάλυσης δεν είναι απόλυτα τυποποιημένες. Οι μέθοδοι αντιμετώπισης των προβλημάτων που προκύπτουν είναι άμεσα εξαρτημένες από το είδος των προβλημάτων και την προσωπική προοπτική και εμπειρία των αναλυτών. Τα διαθέσιμα εργαλεία και οι διάφορες τεχνικές ανάλυσης χρησιμοποιούνται με διάφορους τρόπους από τον κάθε αναλυτή.

Στην προμελέτη του συστήματος υπάρχουν καταγραμμένες και εγκεκριμένες από τον πελάτη (δηλαδή εκείνον για τον οποίο αναπτύσσεται το σύστημα), οι γενικές αρχές που θα διέπουν το νέο σύστημα, χωρίς να αναφέρονται εξειδικευμένες λεπτομέρειες του τύπου "τι λογισμικό ανάπτυξης θα χρησιμοποιηθεί", "τι οθόνες εισαγωγής στοιχείων θα υπάρχουν", "τι είδους αναφορές (εκτυπώσεις) θα παράγονται" κ.ο.κ.

Στη φάση της ανάλυσης πρέπει να συγκεντρωθούν πληροφορίες, οι οποίες θα οδηγήσουν στη σύνταξη συγκεκριμένων οδηγιών προς τις ομάδες που θα αναλάβουν την τελική σχεδίαση και τη δημιουργία των προγραμμάτων, ώστε το έργο τους να είναι ευκολότερο.

Τα ερωτήματα – προβλήματα, τα οποία πρέπει να αντιμετωπισθούν κατά τη φάση αυτή, δεν είναι τυποποιημένα και ποικίλλουν ανάλογα με το είδος, την πολυπλοκότητα και το μέγεθος του υπό κατασκευή συστήματος. Για παράδειγμα, αλλιώς αντιμετωπίζεται το πρόβλημα της ανάλυσης ενός συστήματος διαχείρισης βιντεολέσχης και αλλιώς αναλύονται οι απαιτήσεις του πληροφοριακού συστήματος μιας εταιρίας, που πρέπει να περιλαμβάνει λογιστήριο, αποθήκη, υποσυστήματα παραγγελιών, προμηθευτών, προσωπικού κ.ο.κ.

Είναι συνεπώς σαφές, ότι η ανάλυση είναι μια δυναμική και ευέλικτη διαδικασία, η οποία πρέπει κάθε φορά να προσαρμόζεται στις ανάγκες του προς κατασκευή συστήματος, χωρίς να υπάρχει μέχρι σήμερα κάποια αυστηρή τυποποίηση της όλης πορείας. Σε αδρές γραμμές πάντως, θα μπορούσε κανείς να διακρίνει δύο κύριες φάσεις στη διαδικασία ανάλυσης, όπως σχηματικά παρουσιάζονται στο **σχήμα 5.1.**



Σχήμα 5.1. - Τα δύο βασικά στάδια της ανάλυσης συστημάτων.

Παράδειγμα: Ας υποθέσουμε ότι κάποια επιχείρηση ενοικίασης βιντεοταινιών (βιντεολέσχη), αποφασίζει να παραγγείλει σε κάποιο οίκο ανάπτυξης λογισμικού (*software house*) ένα πρόγραμμα, που θα παρακολουθεί και θα ελέγχει με χρήση ηλεκτρονικού υπολογιστή, όλο τον κύκλο των δραστηριοτήτων της.

Οι αρμόδιοι για την επικοινωνία με τους πελάτες υπάλληλοι του οίκου ανάπτυξης λογισμικού συζητούν κατ' αρχήν με τον υπεύθυνο της επιχείρησης, τι περίπου θα πρέπει να κάνει η ζητούμενη εφαρμογή. Σκοπός τους είναι να διερευνήσουν την πιθανότητα να υπάρχει κάποια ήδη έτοιμη εφαρμογή, που θα μπορούσε να καλύψει τις ανάγκες της συγκεκριμένης επιχείρησης. Αν δε γίνει δυνατός ο εντοπισμός κατάλληλου έτοιμου λογισμικού, αποφασίζεται η παραγγελία ενός συστήματος προσαρμοσμένου στις ανάγκες του πελάτη.

Στην περίπτωση αυτή γίνεται αρχικά η προμελέτη, εγκρίνεται από τον πελάτη και στη συνέχεια αναλαμβάνει η ομάδα αναλυτών να προχωρήσει σε βάθος την ανάλυση των απαιτήσεων, που προβάλλουν από τη φύση της εφαρμογής.

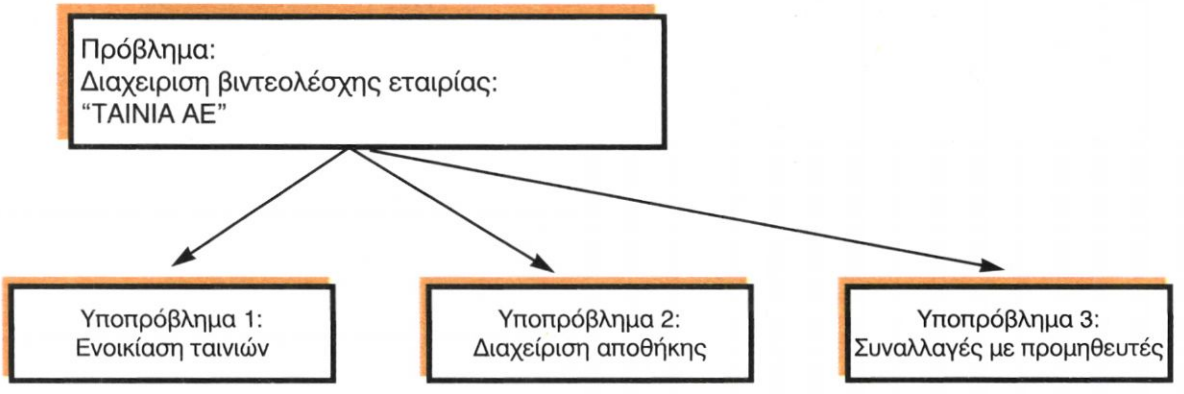
Το πρόβλημα που τίθεται αρχικά για το συγκεκριμένο παράδειγμα της βιντεολέσχης, που αναφέρθηκε παραπάνω, έχει ως εξής: Να κατασκευαστεί μια εφαρμογή ηλεκτρονικού υπολογιστή η οποία, σύμφωνα με όσα καθορίστηκαν στην προμελέτη, θα παρέχει στο χρήστη τις εξής δυνατότητες:

- Παρακολούθηση των ενοικιαζόμενων βιντεοκασετών στους πελάτες (Πότε νοίκιασε κάποιος μια κασέτα, πότε την επέστρεψε, ποιος είναι ο τίτλος της κασέτας, τα στοιχεία του πελάτη, έκδοση αποδείξεων κ.ο.κ.).
- Διαχείριση της αποθήκης, όπου φυλάσσονται οι ταινίες (καταχώρηση νέων τίτλων, πόσα αντίγραφα υπάρχουν ανά τίτλο, καταχώρηση κάθε είδους μεταβολών, όπως αλλαγές αποθεμάτων, διαγραφές τίτλων κ.ο.κ).
- Παρακολούθηση δοσοληψιών με προμηθευτές (παραγγελίες σε τίτλους, κατάσταση υπολοίπου κ.ο.κ).

ΠΑΡΑΤΗΡΗΣΗ

Το αρχικό πρόβλημα της δημιουργίας του προγράμματος για τη βιντεολέσχη αναλύθηκε σε τρία άλλα, πολύ πιο συγκεκριμένα, τα οποία θα αποτελέσουν στα επόμενα στάδια, υποσυστήματα του προς δημιουργία μηχανογραφικού συστήματος.

Ας υποτεθεί ότι η βιντεολέσχη που επιθυμεί την μηχανογράφησή της λέγεται "ΤΑΙΝΙΑ Α.Ε.". Στο **σχήμα 5.2**. παρουσιάζεται η ανάλυση του αρχικού προβλήματος στα τρία υποπροβλήματα, που περιγράψαμε παραπάνω:



Σχήμα 5.2. – Βασικό πρόβλημα και υποπροβλήματα, για το προς κατασκευή μηχανογραφικό σύστημα της εταιρίας "ΤΑΙΝΙΑ Α.Ε".

Η ανάλυση του βασικού προβλήματος σε υποπροβλήματα, εργασία η οποία λίγο ως πολύ γίνεται και στην προμελέτη, είναι αυτή που μας εισάγει στο δεύτερο και ουσιαστικότερο στάδιο της ανάλυσης συστημάτων, που είναι η λεπτομερής ανάλυση των απαιτήσεων συστήματος – χρήστη. Σε αυτό το στάδιο κάθε υποπρόβλημα θα διευκρινιστεί πλήρως ως προς τις απαιτήσεις του, ώστε στη φάση του σχεδιασμού να παραδοθεί απόλυτα προσδιορισμένο.

Σημείωση:

Το παράδειγμα της οργάνωσης γραφείου έκδοσης αεροπορικών εισιτηρίων του Τετραδίου Εργασίας προσφέρεται για εξοικείωση με τις βασικές έννοιες της ανάπτυξης ενός νέου συστήματος. Ειδικά η 1η δραστηριότητα που προτείνεται, είναι ιδιαίτερα χρήσιμη στο να εισάγει στην προβληματική του χωρισμού σε υποπροβλήματα του συγκεκριμένου προβλήματος, που ο αναλυτής καλείται να αντιμετωπίσει, στην πορεία προς την ανάπτυξη του αυτοματοποιημένου συστήματος, που θα υποστηρίξει τη λειτουργία του γραφείου.

5.2. ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ ΠΟΥ ΠΡΟΚΥΠΤΟΥΝ ΑΠΟ ΤΗΝ ΑΝΑΛΥΣΗ ΤΟΥ ΒΑΣΙΚΟΥ ΣΕ ΕΠΙΜΕΡΟΥΣ ΠΡΟΒΛΗΜΑΤΑ

Ορισμός

Σαν **απαίτηση** στη γλώσσα της ανάλυσης συστημάτων, θεωρείται μια λεπτομερής περιγραφή κάποιου συγκεκριμένου μέρους ή λειτουργίας του συστήματος. Η απαίτηση, όπως τουλάχιστον την αντιλαμβανόμαστε σε αυτή τη φάση, εστιάζεται στο τι ανάγκες πρέπει να ικανοποιήσουμε και όχι στο πώς ακριβώς θα γίνει αυτό.

Οι απαιτήσεις αναφέρονται στους στόχους που θέτει ένα σύστημα, χωρίς, όμως, να περιλαμβάνουν λεπτομέρειες σχετικά με την υλοποίηση του συστήματος, η οποία ακολουθεί στη συνέχεια.

Ορισμός

Σαν **ανάλυση απαιτήσεων**, ορίζεται ο προσδιορισμός της φύσης του προβλήματος του πελάτη. Εστιάζεται στον πελάτη και στο πρόβλημα και όχι στην υλοποίηση της λύσης του.

Από πολλούς αναλυτές θεωρείται το δυσκολότερο μέρος της ανάλυσης συστημάτων, αφού το να εκμαιεύσει κανείς από το χρήστη τις πραγματικές του ανάγκες και από αυτές να δώσει μια σαφή εικόνα του συστήματος που πρέπει να κτιστεί τελικά, απαιτεί όχι μόνο γνώσεις, αλλά και ταλέντο στην ανθρώπινη επικοινωνία, καθώς και δυνατότητα συνεκτίμησης όλων των στοιχείων που εμπλέκονται στο πρόβλημα (οικονομικές δυνατότητες, απαιτούμενος εξοπλισμός και προσωπικό από την πλευρά του πελάτη κ.ο.κ).

5.2.1. Κατηγοριοποίηση των απαιτήσεων σε σχέση με τη λειτουργικότητα τους

Οι απαιτήσεις μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες ανάλογα με τη λειτουργικότητά τους:

- **Λειτουργικές απαιτήσεις.** Περιγράφουν τι είδους επαφή πρέπει να υπάρχει ανάμεσα στο σύστημα (την εφαρμογή) και το περιβάλλον του. Δηλαδή, πώς το σύστημα πρέπει να αποκρίνεται όταν δέχεται συγκεκριμένα δεδομένα, τι είδους αποτελέσματα πρέπει να παράγει ανάλογα με τα εισερχόμενα αιτήματα κ.ο.κ.

Παράδειγμα: Όταν μέσω της κάρτας ενός πελάτη της βιντεολέσχης, πληκτρολογηθεί ο κωδικός της ταινίας που ο πελάτης επιθυμεί να νοικιάσει, είναι ανάγκη να ελεγχθούν οι πληροφορίες του υποσυστήματος της αποθήκης, ώστε να φανεί αν υπάρχουν διαθέσιμα αντίτυπα της βιντεοταινίας τη δεδομένη στιγμή. Αυτή η απαίτηση πρέπει να έχει καθοριστεί σαφώς στη φάση της ανάλυσης και υπάγεται στις λειτουργικές απαιτήσεις.

- **Μη λειτουργικές απαιτήσεις.** Αποτελούν τους αναγκαστικούς περιορισμούς, που η πραγματικότητα θέτει, περιορίζοντας έτσι κάποιες από τις διαθέσιμες επιλογές στην ανάπτυξη του συστήματος.

Τέτοιες μη λειτουργικές απαιτήσεις είναι οι περιορισμοί που μπορεί να υπάρχουν σε σχέση με το πλήθος των διαθέσιμων ηλεκτρονικών υπολογιστών, την ισχύς τους, το λογισμικό που είναι προεγκατεστημένο σε αυτούς, τυχόν χρονικοί περιορισμοί που υφίστανται, προβλήματα περιορισμένου κοστολογίου για το έργο κ.λ.π.

Παράδειγμα:

Έστω, ότι ο ιδιοκτήτης της "ΤΑΙΝΙΑ Α.Ε.", έχει στη διάθεσή του δύο προσωπικούς υπολογιστές και θέλει να χρησιμοποιείται η εφαρμογή του ταυτόχρονα και από τους δύο. Δεν θέλει, όμως, να πληρώσει την αυξημένη τιμή που θα κοστίσει η εφαρμογή, ώστε να λειτουργεί σε τοπικό δίκτυο, ούτε να πληρώσει το κόστος του εξοπλισμού δικτύωσης (κάρτες δικτύου, καλώδια). **Αυτό συνιστά μια μη λειτουργική απαίτηση**, την οποία όμως ένας καλός αναλυτής μπορεί, με σωστό χειρισμό προς την πλευρά του πελάτη, να ξεπεράσει.

5.2.2. Γενικά χαρακτηριστικά των απαιτήσεων

Στο κεφάλαιο 4 περιγράφηκε ο τρόπος με τον οποίο συγκεντρώνονται οι πληροφορίες που προσδιορίζουν το σύστημα με τη χρήση ερωτηματολογίων και συνεντεύξεων. Στη συνέχεια θα

εξεταστούν οι τρόποι, με τους οποίους γίνεται η ανάλυση των πληροφοριών που έχουν συγκεντρωθεί.
Όπως αναφέρθηκε και στο κεφάλαιο 4 η ανάλυση των απαιτήσεων ενός συστήματος θα πρέπει να προσδιορίζει:

- **Τις ροές πληροφοριών από και προς το σύστημα,** δηλαδή, τι είδους πληροφορίες θα εισάγονται στο σύστημα, ενώ παράλληλα πρέπει να περιγράφεται αναλυτικά τι πληροφορίες θα παράγονται (βγαίνουν) από αυτό και με τι τρόπο (π.χ., μέσα από φόρμες στην οθόνη, ή ως εκτυπώσιμες καταστάσεις;).

- **Τους μετασχηματισμούς των δεδομένων από το σύστημα,** δηλαδή τον τρόπο με τον οποίο το αυτοματοποιημένο σύστημα δίνει στα εισαγόμενα από το χρήστη δεδομένα, μηχανογραφικά σωστό περιεχόμενο έτσι ώστε η διαχείρισή τους, να γίνεται πλέον με τέτοια μορφή.

Παράδειγμα:

Αυτό που ο ιδιοκτήτης της "ΤΑΙΝΙΑ Α.Ε." έχει στο μυαλό του, όταν αναφέρεται σε συναλλαγή με τους προμηθευτές των ταινιών (π.χ., το όνομα του προμηθευτή, τους τίτλους ταινιών που θέλει να παραγγείλει, τον αριθμό αντιτύπων ανά τίτλο, τον οικονομικό διακανονισμό κ.ο.κ.), πρέπει να μετατραπεί σε συγκεκριμένης μορφής εγγραφή, κατάλληλη για το μηχανογραφικό σύστημα που θα δημιουργηθεί.

Δηλαδή, σε κάθε προμηθευτή θα αντιστοιχεί μια εγγραφή καταχωρημένη στα μηχανογραφικά αρχεία του συστήματος, η οποία θα περιλαμβάνει συγκεκριμένες πληροφορίες, όπως: κωδικό προμηθευτή, ονομασία επιχείρησης του προμηθευτή, έδρα επιχείρησης, αριθμό τηλεφώνου, αριθμό fax, e-mail διεύθυνση (αν υπάρχει), χρεωστικό υπόλοιπο συναλλαγών κ.λ.π.

Με ανάλογο τρόπο θα σχεδιαστεί ξεχωριστή εγγραφή που θα αντιστοιχεί σε κάθε είδος, όπως αυτό θα περνά στην αποθήκη της επιχείρησης, άλλη εγγραφή για τη διαχείριση της καρτέλας συναλλαγής με τους πελάτες κ.ο.κ.

Με αυτή τη λογική, ένα αυτοματοποιημένο σύστημα εκφράζει το πέρασμα από μία ασαφή ή όχι καλά οργανωμένη υλοποίηση μιας εργασίας, στη δομημένη και αυτοματοποιημένη λειτουργία της, που υποστηρίζεται από συστήματα ηλεκτρονικών υπολογιστών.

5.2.3. Κριτήρια διαμόρφωσης των απαιτήσεων

Όταν ο αναλυτής καθορίζει τις απαιτήσεις σε συνεργασία με τον πελάτη, πρέπει να τις διαμορφώσει έτσι, ώστε να ικανοποιούν τα πιο κάτω κριτήρια:

- ➢ Ακρίβεια στον ορισμό τους.
- ➢ Απουσία αλληλοαναιρούμενων ή αλληλοεπικαλυπτόμενων απαιτήσεων.
- ➢ Πληρότητα, δηλαδή προσδιορισμό όλων των δυνατών περιεχομένων.
- ➢ Οι απαιτήσεις πρέπει να είναι ρεαλιστικές και όχι ουτοπικές.
- ➢ Πρέπει να ικανοποιούν τις πραγματικές ανάγκες του πελάτη και όχι να του προσφέρουν επιλογές που δεν χρειάζεται.
- ➢ Πρέπει να είναι επαληθεύσιμες, δηλαδή να μπορούν να υπαχθούν σε λογικούς ελέγχους επαλήθευσης, στους οποίους οφείλει ο αναλυτής να τις υποβάλει, ώστε να διαπιστώσει αν και σε ποιο βαθμό αντιστοιχούν στις ανάγκες του συστήματος.

Ο σωστός, πλήρης και ακριβής προσδιορισμός των απαιτήσεων είναι κρίσιμο ζήτημα, στο οποίο ο αναλυτής αποδίδει ιδιαίτερη έμφαση από την αρχή, γιατί είναι βέβαιο ότι θα κερδίσει χρόνο και χρήμα αποφεύγοντας την εκ των υστέρων αναθεώρηση μεγάλου μέρους της εργασίας, στην περίπτωση, που κατά τη φάση των ελέγχων, διαπιστωθεί ότι υλοποιήθηκε σύστημα βασισμένο σε λάθος εκτιμήσεις.

5.2.4. Κατηγορίες απαιτήσεων ανάλογα με το πεδίο προσδιορισμού τους

Στον προσδιορισμό των απαιτήσεων ενός συστήματος περιλαμβάνεται και η περιγραφή οποιουδήποτε στοιχείου αφορά στην επικοινωνία του συστήματος με το περιβάλλον του. Η αναλυτική αναφορά στα στοιχεία που συνθέτουν το περιβάλλον ενός συστήματος ξεφεύγει από τους στόχους του παρόντος βιβλίου. Είναι, όμως, σκόπιμο στο σημείο αυτό να γίνει μία συνοπτική αναφορά στους σημαντικότερους παράγοντες από τους οποίους εξαρτάται και προσδιορίζεται το περιβάλλον αυτό, χωρίς σε καμία περίπτωση να απαιτείται η απομνημόνευσή τους, αλλά μόνο για να σχηματιστεί πληρέστερη εικόνα.

- Φυσικό περιβάλλον.
  - ➢ Πού πρέπει να εγκατασταθεί ο εξοπλισμός από άποψη χωροταξίας.
  - ➢ Περιβαλλοντολογικοί περιορισμοί (σκόνη, προβλήματα θερμοκρασίας κ.α.).

- Επικοινωνία με το σύστημα.
  - ➢ Εισερχόμενα δεδομένα.
  - ➢ Εξερχόμενα δεδομένα.
  - ➢ Μορφοποίηση δεδομένων.
  - ➢ Μαγνητικά μέσα για αποθήκευση δεδομένων, για επικοινωνία με το περιβάλλον, ή για τήρηση αντιγράφων ασφάλειας (backup).

- Ανθρώπινος παράγοντας.
  - ➢ Χρήστες συστήματος.
  - ➢ Απαιτήσεις εκπαίδευσης των χρηστών.
  - ➢ Ευκολία χρήσης και κατανόησης του συστήματος.

➢ Δυνατότητες προφύλαξης του συστήματος από κακή χρήση.

• Λειτουργικότητα.
➢ Τι ακριβώς πρέπει να κάνει το σύστημα.
➢ Πότε το σύστημα πρέπει να βρίσκεται σε λειτουργία.
➢ Πότε και πώς πρέπει να γίνονται αλλαγές.
➢ Περιορισμοί στην ταχύτητα εκτέλεσης και στην απόκριση του συστήματος.

• Υλικό τεκμηρίωσης.
➢ Απαιτούμενα εγχειρίδια βοήθειας του χρήστη.
➢ Δυνατότητα βοήθειας άμεσης επικοινωνίας *(on-line)*, μέσα από τη χρήση του συστήματος.

• Δεδομένα.
➢ Μορφή των δεδομένων εισόδου - εξόδου.
➢ Ακρίβεια στη μορφή των δεδομένων.
➢ Αριθμός δεκαδικών ψηφίων στα μη ακέραια δεδομένα.
➢ Συχνότητα τροποποιήσεων στα δεδομένα.

• Πόροι.
➢ Πόροι απαραίτητοι για το κτίσιμο, τη χρήση και τη διαχείριση του συστήματος.
➢ Προσόντα που απαιτούνται για τους συμμετέχοντες στην ανάπτυξη του συστήματος.
➢ Απαιτούμενοι χώροι φυσικής αποθήκευσης δεδομένων στον υπολογιστή.
➢ Απαιτήσεις ως προς την ποιότητα του περιβάλλοντος εργασίας των χρηστών του συστήματος και τις συνθήκες κλιματισμού και τροφοδοσίας των μηχανημάτων με τα οποία θα υλοποιηθεί το νέο σύστημα.
➢ Χρονοπρογραμματισμός της φάσης υλοποίησης.
➢ Διαθέσιμοι οικονομικοί πόροι.

• Ασφάλεια συστήματος.
➢ Καθορισμός επιπέδων πρόσβασης στις επιμέρους λειτουργίες του συστήματος.
➢ Είδη και εξοπλισμός για τήρηση αντιγράφων ασφάλειας των δεδομένων (backup).
➢ Δυνατότητα προστασίας των χώρων, όπου είναι εγκατεστημένος ο εξοπλισμός, από φωτιά και κλοπή.

• Διασφάλιση ποιότητας.
➢ Απαιτήσεις αξιοπιστίας.
➢ Δυνατότητες επισήμανσης και αντιμετώπισης τυχόν λαθών, που εκ των υστέρων θα εντοπιστούν.
➢ Κατά προσέγγιση εκτίμηση μέσου χρόνου βλαβών στον εξοπλισμό του συστήματος *(mean time between failure)*.
➢ Εκτίμηση και απαιτήσεις για χρόνο αποκατάστασης της λειτουργίας του εξοπλισμού μετά από σοβαρή βλάβη.
➢ Απαιτήσεις για μετατροπές και επεκτάσεις, που τυχόν θα χρειαστούν στην πορεία ζωής του συστήματος και για εύκολη μεταφορά των προγραμμάτων και δεδομένων του συστήματος σε ένα άλλο υπολογιστικό περιβάλλον *(portability)*.

Ανάλογα με την πολυπλοκότητα του συστήματος, θα πρέπει να δοθούν σαφείς απαντήσεις, είτε σε όλες τις κατηγορίες των πιο πάνω απαιτήσεων, είτε σε κάποιες από αυτές. Άλλωστε η ανάλυση συστημάτων είναι, όπως έχει εξηγηθεί παραπάνω, μια πορεία ευέλικτη και προσαρμοσμένη πάντα στις ανάγκες των συγκεκριμένων προβλημάτων που έχει να αντιμετωπίσει.

5.2.5. Απεικόνιση της ροής πληροφοριών στο σύστημα

Θεωρείστε το σχολείο σαν σύστημα. Μπορείτε να αναφέρετε μερικά εισερχόμενα δεδομένα; Σε ποια διαδικασία μετατροπής υπόκεινται αυτά; Γνωρίζετε μερικά από τα εξερχόμενα δεδομένα του συστήματος;

Εξετάζοντας ένα οποιοδήποτε σύστημα, αυτοματοποιημένο είτε όχι, θα διαπιστώσει κανείς ότι στον πυρήνα του είναι ενσωματωμένος πάντα, κάποιος μηχανισμός επεξεργασίας πληροφοριών. Ο μηχανισμός αυτός μετατρέπει τα εισερχόμενα δεδομένα (αναφερόμενα και σαν δεδομένα εισόδου - input) που δέχεται στη είσοδό του, σε εξερχόμενα δεδομένα (αναφερόμενα και σαν δεδομένα εξόδου - output), δηλαδή σε πληροφορίες κάποιας άλλης μορφής, οι οποίες έχουν προκύψει μετά από επεξεργασία μέσα στο σύστημα. Αυτού του είδους η ροή πληροφοριών από και προς το σύστημα, πρέπει να αναλυθεί στο μέγιστο δυνατό βαθμό, ώστε η τελική μορφή του συστήματος να ικανοποιεί, όλες τις βασικές απαιτήσεις του πελάτη.

Για την καλύτερη και περισσότερο εποπτική απεικόνιση της ροής των πληροφοριών και για την αποτελεσματικότερη ανάλυσή τους, χρησιμοποιούνται συγκεκριμένα πρότυπα (μοντέλα) απεικόνισης, γνωστά σαν πρότυπα ροής πληροφοριών ή δεδομένων *(data-flow models)*.

Τα πρότυπα αυτού του είδους χρησιμοποιούν σαν εργαλεία ειδικά διαγράμματα, τα διαγράμματα ροής πληροφοριών ή δεδομένων, τα οποία για λόγους συντομίας συνήθως αναφέρονται σαν ΔΡΔ.

5.2.6. Διαγράμματα ροής δεδομένων

Τα διαγράμματα ροής δεδομένων (ΔΡΔ) *(data-flow diagrams - DFD)*, απεικονίζουν σχηματικά τις σχέσεις των δεδομένων με τις διεργασίες οι οποίες τα επεξεργάζονται, με σκοπό τη σύνθεση ενός προτύπου που παριστάνει ένα μέρος, ή και ολόκληρο το σύστημα. Τα διαγράμματα ροής δεδομένων βοηθούν τον αναλυτή στην περιγραφή, τόσο του νέου συστήματος που πρόκειται να αναπτυχθεί, όσο και του υπάρχοντος, το οποίο πρόκειται να αντικατασταθεί από το νέο. Αν κάποιος γνωρίζει στοιχειωδώς το σύστημα που περιγράφει ένα ΔΡΔ, μπορεί αρκετά εύκολα να κατανοήσει το πρότυπο που απεικονίζεται στο διάγραμμα, χωρίς να απαιτούνται ιδιαίτερα λεπτομερείς περιγραφές. Κάθε τυπικό διάγραμμα ΔΡΔ σχηματίζεται από κάποια απλά σχήματα - σύμβολα, καθένα από τα οποία έχει την ξεχωριστή του σημασία.

Στον παρακάτω **πίνακα 5.1.** απεικονίζονται τα πιο συνηθισμένα σύμβολα, που χρησιμοποιούνται σε ένα ΔΡΔ, με τη σημασία τους. Επισημαίνεται ότι σε κάθε συμβολισμό αντιστοιχούν περισσότερα από ένα σχήματα από τα οποία ο αναλυτής επιλέγει ένα, ανάλογα με τις προτιμήσεις του.

Όνομα	Σύμβολο	Σημασία
Διεργασία (process)	(Από κάποιους χρησιμοποιείται ο κύκλος, από άλλους η έλλειψη. Η πιο σύγχρονη εκδοχή είναι το τρίτο σχήμα, γνωστό σαν φυσαλίδα.	Διαδικασία επεξεργασίας δεδομένων.
Ροή δεδομένων (data - flow)		Ροή δεδομένων μεταξύ των άλλων συμβόλων του διαγράμματος.
Εξωτερική οντότητα (external entity) - αφετηρία ή προορισμός	(Κάποιοι αναλυτές το σχεδιάζουν Ορθογώνιο)	Συμβολίζει μια πηγή προέλευσης δεδομένων ή κάποιον προορισμό δεδομένων.
Αποθήκη δεδομένων (data store)	(Η συνηθέστερη απεικόνιση).	Αρχείο ή Βάση, που αποθηκεύονται τα δεδομένα του συστήματος.

Πίνακας 5.1. – Σύμβολα σχεδιασμού διαγραμμάτων ροής δεδομένων.

- **Διεργασία - ροή δεδομένων.**

Ορισμός

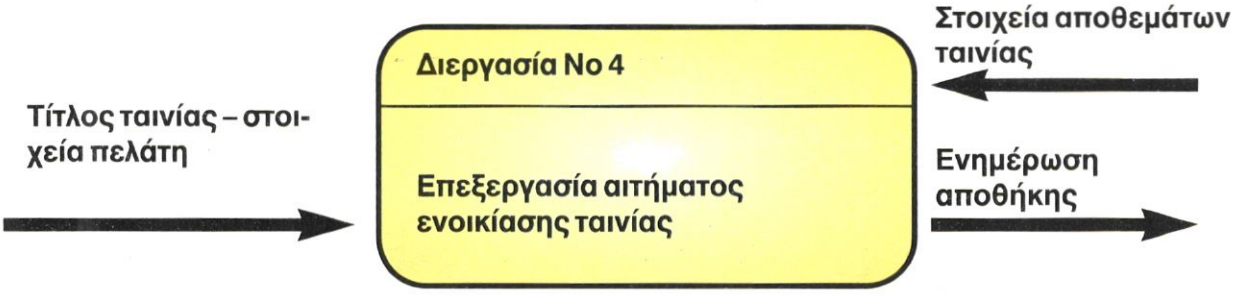
Διεργασία στη γλώσσα της ανάλυσης συστημάτων είναι μια διαδικασία, που μετατρέπει εισερχόμενα δεδομένα, σε παραγόμενες, μετά από επεξεργασία, πληροφορίες. Η απεικόνιση αυτής της διαδικασίας γίνεται με τα σύμβολα που απεικονίζονται στον **πίνακα 5.1.**

Μετατροπή των εισερχόμενων στη διεργασία δεδομένων σε εξερχόμενες πληροφορίες σημαίνει ότι τα δεδομένα μπορεί να συνδεθούν μεταξύ τους σε διάφορες μορφές, να ταξινομηθούν με ορισμένη σειρά, ή να συνθέσουν μια εκτυπώσιμη αναφορά, κατανοητή στους χρήστες του συστήματος.

Στο **σχήμα 5.3.** που ακολουθεί παρουσιάζεται η ροή δεδομένων από και προς τη διεργασία, η οποία περιγράφεται σαν "επεξεργασία αιτήματος ενοικίασης ταινίας". Θα μπορούσε να αναφέρεται στην υποθετική βιντεολέσχη, την οποία έχουμε ήδη γνωρίσει σαν "ΤΑΙΝΙΑ Α.Ε.", αποτελώντας ένα ελάχιστο μέρος ενός αναλυτικού διαγράμματος ροής δεδομένων, που παρουσιάζει τις οντότητες, τα αρχεία και τις διαδικασίες, τα οποία θα αποτελέσουν το σκελετό του αντίστοιχου αυτοματοποιημένου συστήματος.

Η μια ροή (η εισερχόμενη από αριστερά) συμβολίζει τα δεδομένα (όχι με λεπτομέρειες), τα οποία συνιστούν παραγγελία ενός πελάτη για ενοικίαση ταινίας. Η εξερχόμενη ροή συμβολίζει κάποιες πληροφορίες, που θα αξιοποιηθούν από το υποσύστημα αποθήκης το οποίο θα ενημερώσουν για την ενοικίαση της ταινίας, ώστε να καταγράψει μείωση κατά ένα των διαθέσιμων αντιτύπων του τίτλου αυτού. Όμως η διεργασία αυτή οφείλει, πριν την ενοικίαση της ταινίας που ζητείται, να γνωρίζει αν υπάρχει διαθέσιμο αντίτυπο. Η πληροφορία αυτή θα ζητηθεί από το υποσύστημα της αποθήκης και παριστάνεται με την ροή που εισέρχεται από τα δεξιά στη διεργασία.

Η ίδια η διεργασία απεικονίζεται με το πιο διαδεδομένο στη σύγχρονη ανάλυση σύμβολο φυσαλίδας, όπου στο επάνω μέρος υπάρχει ένα συμβολικό για το διάγραμμα όνομα και στο κάτω μέρος υπάρχει η αναλυτική περιγραφή της διεργασίας.



Σχήμα 5.3. – Διεργασία και ροές δεδομένων.

• Εξωτερική οντότητα-εσωτερική οντότητα.

Ο όρος **εξωτερική οντότητα** αναφέρεται σε υπαρκτές οντότητες, που επικοινωνούν με το υπό μελέτη σύστημα, βρίσκονται έξω από αυτό και ανταλλάσσουν με αυτό στοιχεία. Χρησιμοποιούνται από τις διεργασίες του συστήματος σαν δεξαμενές εξόδου - εισόδου δεδομένων.

Ορισμός

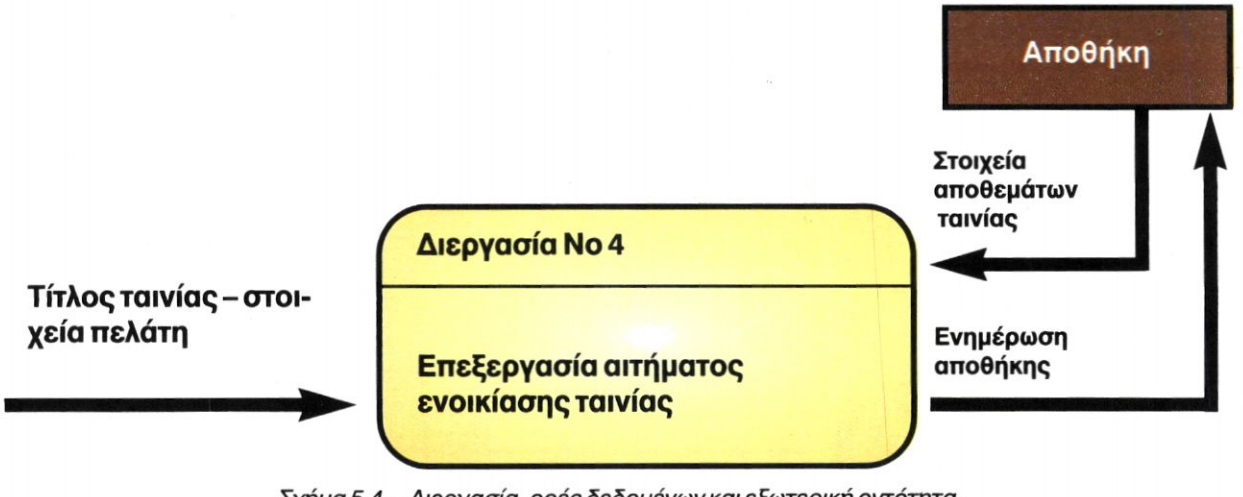
Το γεγονός ότι οι οντότητες αυτές εμφανίζονται στα διαγράμματα που σχεδιάζονται για το σύστημα, πολλές φορές ξεγελά και δημιουργεί την ψευδή εντύπωση ότι αποτελούν μέρος του. Εξωτερική οντότητα, από λογική άποψη, μπορεί να είναι ένα φυσικό πρόσωπο ή ομάδα ανθρώπων, για παράδειγμα: ένας οργανισμός, μια κρατική υπηρεσία, ένα τμήμα στα όρια της ίδιας επιχείρησης το οποίο όμως βρίσκεται εκτός του άμεσου ελέγχου του συστήματος, το οποίο απεικονίζεται σε ένα διάγραμμα. Μερικές φορές εξωτερική οντότητα μπορεί να αποτελεί κάποιο ξεχωριστό αυτοματοποιημένο σύστημα.

Στο απλό παράδειγμα της βιντεολέσχης που έχει παρουσιαστεί στα προηγούμενα, φάνηκε να εντάσσουμε τη διαχείριση της αποθήκης και την παρακολούθηση των δοσοληψιών με τους προμηθευτές στο ίδιο το σύστημα ενοικίασης βιντεοταινιών. Επειδή αυτού του μεγέθους επιχειρήσεις δεν χαρακτηρίζονται από ιδιαίτερη πολυπλοκότητα, στην πράξη μπορεί να θεωρηθούν οι προμηθευτές και η αποθήκη, όχι σαν ξεχωριστά συστήματα με αυτόνομη λειτουργικότητα, αλλά σαν εσωτερικές οντότητες στο σύστημα της βιντεολέσχης. Οι εσωτερικές οντότητες ενός συστήματος παρουσιάζουν λιγότερο ενδιαφέρον για την ανάλυση και περισσότερο για την οργάνωση των δεδομένων και το σχεδιασμό.

Οι διεργασίες, οι ροές δεδομένων, οι εξωτερικές οντότητες και οι αποθήκες δεδομένων αποτελούν βασικά στοιχεία που απεικονίζει ένα διάγραμμα ροής δεδομένων.

Σε μια πιο απαιτητική, όμως, μηχανογράφηση της επιχείρησης, θα μπορούσαν να θεωρηθούν ξεχωριστά συστήματα, τα οποία επικοινωνούν ανταλλάσσοντας στοιχεία με το σύστημα διακίνησης ταινιών. Στην περίπτωση αυτή θα ήταν περισσότερο απαιτητικά στο σχεδιασμό τους και θα αποτελούσαν εξωτερικές οντότητες στο διάγραμμα ροής δεδομένων του συστήματος.

Το **σχήμα 5.4** που ακολουθεί αποτελεί παραλλαγή του σχήματος 5.3, και εμφανίζει την αποθήκη σαν εξωτερική οντότητα.



Σχήμα 5.4. – Διεργασία, ροές δεδομένων και εξωτερική οντότητα.

• **Αποθήκη δεδομένων.**

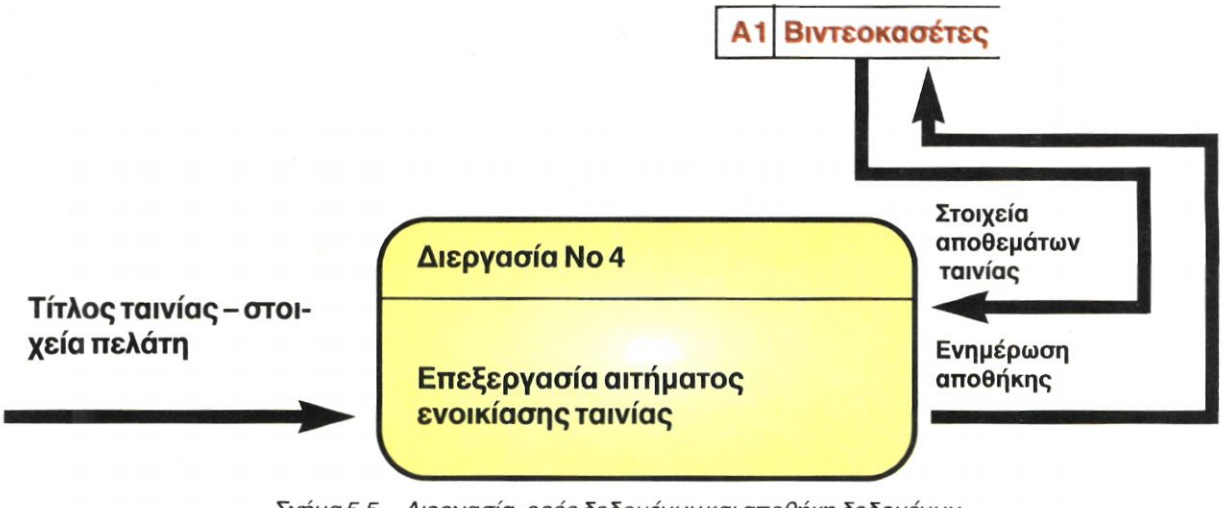
Ορισμός

Η **αποθήκη δεδομένων** αντιπροσωπεύει μια συλλογή από πακέτα δεδομένων. Συμβολίζεται, όπως αναφέρθηκε στο σχετικό πίνακα, με δύο τρόπους (ανοιχτό προς τα δεξιά ορθογώνιο ή ένα ζευγάρι παράλληλων διπλών γραμμών). Για τους αναλυτές αυτοματοποιημένων συστημάτων, οι αποθήκες δεδομένων δεν είναι τίποτα άλλο από παραδοσιακά αρχεία και βάσεις δεδομένων.

Όταν αναφερόμαστε σε μη αυτοματοποιημένα συστήματα, αποθήκες δεδομένων μπορεί να είναι, για παράδειγμα, ένα ράφι με φάρμακα κάποιας κατηγορίας σε ένα φαρμακείο, οι θυρίδες που μαζεύονται τα έγγραφα που πρωτοκολλούνται σε μία δημόσια υπηρεσία, η καρτελοθήκη με τα στοιχεία των μαθητών ενός σχολείου ή φροντιστηρίου, τα λογιστικά βιβλία μιας επιχείρησης κ.λ.π.

Οι αποθήκες δεδομένων στο ΔΡΔ επικοινωνούν με το υπόλοιπο σύστημα με ροές. Οι εισερχόμενες σε μια αποθήκη δεδομένων ροές δείχνουν ότι η διεργασία από την οποία ξεκινούν τροποποιεί την αποθήκη, προσθέτοντας, τροποποιώντας, ή διαγράφοντας δεδομένα. Οι εξερχόμενες από μια αποθήκη ροές, εισέρχονται σε κάποια διεργασία που χρησιμοποιεί τα δεδομένα που μεταφέρουν, είτε αυτούσια, είτε σε σύνθεση με άλλα δεδομένα.

Στο **σχήμα 5.5** χρησιμοποιείται για την αποθήκη δεδομένων το σύμβολο του ανοικτού δεξιά ορθογωνίου. Στο πρώτο τμήμα του συμβόλου αναγράφεται μια κωδικοποιημένη ονομασία της αποθήκης, κατάλληλη για σύντομη αναφορά (Α1). Στο δεύτερο, ένα όνομα που περιγράφει το είδος των δεδομένων της αποθήκης (βιντεοκασέτες).



Σχήμα 5.5. – Διεργασία, ροές δεδομένων και αποθήκη δεδομένων.

5.2.7. Γενικές αρχές για την κατασκευή διαγραμμάτων ροής δεδομένων

Η κατασκευή ΔΡΔ είναι πολύ σημαντική διαδικασία για τη σωστή αποτύπωση των αναγκών του συστήματος. Για το λόγο αυτό, βασικό μέλημα του αναλυτή είναι να σχεδιάζει ΔΡΔ, τα οποία να είναι ευανάγνωστα και να προσφέρουν ξεκάθαρη εικόνα των αναγκών και απαιτήσεων του συστήματος. Με τον τρόπο αυτό περιορίζονται σημαντικά οι πιθανότητες παρανοήσεων, όταν τα αποτελέσματα και η τεκμηρίωση της ανάλυσης παραδοθούν στην ομάδα που θα προχωρήσει στον τελικό σχεδιασμό και την υλοποίησή του νέου συστήματος.

Υπάρχουν κάποιες γενικές αρχές, που αν ακολουθηθούν, μπορεί να οδηγήσουν σε καλύτερα και περισσότερο σαφή ΔΡΔ. Οι αρχές αυτές, όπως τις παρουσιάζει ο Edward Yourdon στο κλασσικό βιβλίο του "Modern Structured Analysis", είναι οι εξής:

- Να επιλέγονται ονόματα που να είναι χαρακτηριστικά για τις απεικονιζόμενες διεργασίες, ροές, αποθήκες δεδομένων και εξωτερικές οντότητες. Δεν είναι λογικό, για παράδειγμα, να δοθεί σε μια διεργασία ένα ανθρώπινο όνομα.

- Να υπάρχει κωδικοποίηση ή αρίθμηση των διεργασιών. Συνήθως προτιμάται η αρίθμηση. Π.χ., "διεργασία Νο 3", ή "διεργασία 3.2.1", όταν ένα ΔΡΔ διασπάται σε πολλά ιεραρχικά επίπεδα (ιεραρχικό ΔΡΔ).

- Να δίνεται έμφαση στην αισθητική των διαγραμμάτων. Η εμφάνιση ενός ΔΡΔ παίζει σημαντικό ρόλο στη λειτουργικότητά του. Για το λόγο αυτό, αν διαπιστωθεί ανάγκη επανασχεδιασμού του, ώστε να γίνει αισθητικά καλύτερο, δεν πρέπει να υπάρξει ο παραμικρός δισταγμός, όσες φορές και αν χρειαστεί να γίνει κάτι τέτοιο.

- Να αποφεύγονται οι περίπλοκες συνθέσεις. Το ΔΡΔ, όπως εξηγήθηκε παραπάνω, πρέπει να είναι ευανάγνωστο. Σε περίπτωση που οι ανάγκες απαιτήσουν την κατασκευή ενός μεγάλου και σύνθετου ΔΡΔ, είναι προτιμότερο αυτό να απεικονιστεί τμηματικά σε περισσότερες από μία σελίδες, οι οποίες θα συνδέονται μεταξύ τους με χαρακτηριστικές ενδείξεις. Για παράδειγμα, κάθε σελίδα του διαγράμματος θα έχει κάποιο χαρακτηριστικό αριθμό ή άλλο επιλεγμένο κωδικό και με βέλη, που θα συνοδεύονται από τους κατάλληλους αριθμούς, θα γίνεται η παραπομπή στη σελίδα που ακολουθεί και στη σελίδα που προηγείται.

- Να διασφαλίζεται η εσωτερική συνοχή του διαγράμματος και η συνοχή του με τυχόν άλλα διαγράμματα, με τα οποία συνδέεται. Δεν είναι π.χ., αποδεκτό να υπάρχει κάποια ενδιάμεση ροή, η οποία δείχνει στο κενό, δηλαδή να μη καταλήγει κάπου.

- Πρέπει σε κάθε ΔΡΔ να υπάρχει αρχή και τέλος και όλα τα στοιχεία που το συνθέτουν πρέπει με κάποιο τρόπο να είναι διασυνδεδεμένα μεταξύ τους.

Σημείωση:

Στο αντίστοιχο κεφάλαιο του Τετραδίου Εργασίας σας δίνεται η ευκαιρία να εργαστείτε πάνω στην οργάνωση των δεδομένων ενός συστήματος και στην κατασκευή διαγραμμάτων ροής. Έτσι αποκτούν πρακτικό περιεχόμενο οι πιο πάνω θεωρητικές γνώσεις.

5.2.8. Λεξικό δεδομένων *(data dictionary)*

Όλα τα διαγράμματα που χρησιμοποιούνται στην ανάλυση και το σχεδιασμό ενός συστήματος αποτελούν μια συμβολική αποτύπωση, τόσο των επιμέρους στοιχείων του, όσο και των μεταξύ αυτών υφιστάμενων λειτουργικών διασυνδέσεων. Επίσης, τα διαγράμματα πρέπει να είναι περιεκτικά, κατανοητά, πλήρη και συνοπτικά. Για να επιτευχθούν αυτοί οι στόχοι, απαιτείται η ύπαρξη κάποιας μεθόδου, με την οποία θα διευκρινίζονται και θα τυποποιούνται οι όροι και τα σύμβολα που περιέχονται σε ένα διάγραμμα, ώστε αυτό να γίνεται κατανοητό, ακόμη και από κάποιον που δε συμμετείχε με τρόπο άμεσο στη σχεδίασή του. Η μέθοδος που χρησιμοποιείται ευρύτατα, είναι η δημιουργία ενός λεξικού δεδομένων.

Ορισμός

Το λεξικό δεδομένων *(data dictionary)* είναι ένα ευρετήριο στο οποίο εξηγούνται όλοι οι όροι που χρησιμοποιούνται στη φάση της ανάλυσης και του σχεδιασμού και αναφέρονται σε διεργασίες, ροές, αποθήκες δεδομένων αλλά και στα ίδια τα δεδομένα.

Παράδειγμα:

Η ροή που εμφανίζεται σε προηγούμενο ΔΡΔ (σχήμα 5.5) κωδικοποιημένη σαν "τίτλος ταινίας - στοιχεία πελάτη", δηλώνει με γενικό τρόπο τι είδους πληροφορίες κατευθύνονται στη διεργασία No 4.

Όνομα στοιχείου	*ΚΩΔΙΚΟΣ_ΤΑΙΝΙΑΣ*
Συνώνυμα	ΚΩΔΤ
Τύπος στοιχείου	Αριθμητικό
Επιτρεπτές τιμές	>0 και ακέραιες
Περιγραφή	Μοναδικός κωδικός-ταυτότητα της ταινίας.
Άλλες παρατηρήσεις	

Πίνακας 5.2. – Ενδεικτική καταχώρηση στοιχείου δεδομένων σε ένα λεξικό.

Όνομα διεργασίας	*ΕΝΟΙΚΙΑΣΗ_ΤΑΙΝΙΑΣ*
Συνώνυμα	ΕΝΚΤ
Είσοδος	ΣΤΟΙΧΕΙΑ ΠΕΛΑΤΗ - ΤΑΙΝΙΑΣ
Έξοδος	ΕΝΗΜΕΡΩΣΗ ΑΠΟΘΗΚΗΣ
Περιγραφή	Μόλις ο πελάτης ζητήσει μια ταινία Πληκτρολογείται ο τίτλος, ελέγχεται αν υπάρχει διαθέσιμο αντίτυπο του τίτλου αυτού, αν ναι, ενημερώνεται η καρτέλα του πελάτη ότι νοίκιασε την ταινία αυτή, ενώ παράλληλα ενημερώνεται η αποθήκη ότι τα διαθέσιμα αντίτυπα μειώθηκαν κατά ένα.

Πίνακας 5.3. – Καταχώρηση στοιχείων διεργασίας.

Όνομα ροής δεδομένων	*ΣΤΟΙΧΕΙΑ_ΤΑΙΝΙΑΣ_ΠΕΛΑΤΗ*
Συνώνυμα	ΣΤΤΠ
Περιγραφή	Όλα τα απαραίτητα στοιχεία του πελάτη μαζί με τον τίτλο της ταινίας που θέλει, ή ο κωδικός της, αν είναι γνωστός. Κάποια από τα στοιχεία αυτά είναι απαραίτητα για τη διαδικασία στην οποία εισέρχεται η ροή, κάποια άλλα όχι.
Δομή των δεδομένων που απαρτίζουν τη ροή	ΚΩΔΙΚΟΣ_ΠΕΛΑΤΗ ΟΝΟΜΑΤΕΠΩΝΥΜΟ_ΠΕΛΑΤΗ ΚΩΔΙΚΟΣ_ΤΑΙΝΙΑΣ ΤΙΤΛΟΣ_ΤΑΙΝΙΑΣ ΗΜΕΡΟΜΗΝΙΑ_ΕΝΟΙΚΙΑΣΗΣ ΗΜΕΡΟΜΗΝΙΑ_ΕΠΙΣΤΡΟΦΗΣ ΤΙΜΗ_ΕΝΟΙΚΙΑΣΗΣ

Πίνακας 5.4. – Καταχώρηση στοιχείων ροής.

Όνομα αποθήκης δεδομένων	*ΒΙΝΤΕΟΚΑΣΕΤΕΣ*
Συνώνυμα	VIDEOCS
Περιγραφή	Αποθήκη εγγραφών που αντιστοιχούν στις βιντεοκασέτες
Δομή δεδομένων(εγγραφή) της αποθήκης	ΚΩΔΙΚΟΣ_ΤΑΙΝΙΑΣ ΤΙΤΛΟΣ_ΤΑΙΝΙΑΣ ΑΡΙΘΜ_ΑΝΤΙΤΥΠΩΝ ΠΡΟΜΗΘΕΥΤΗΣ ΗΜΕΡΟΜΗΝΙΑ_ΚΥΚΛΟΦΟΡΙΑΣ ΧΩΡΟΣ_ΑΠΟΘΗΚΕΥΣΗΣ
Ταυτότητα εγγραφής - (Κλειδί εντοπισμού)	ΚΩΔΙΚΟΣ_ΤΑΙΝΙΑΣ
Δευτερεύον κλειδί προσδιορισμού	ΤΙΤΛΟΣ_ΤΑΙΝΙΑΣ

Πίνακας 5.5. – Καταχώρηση στοιχείων αποθήκης δεδομένων.

Οι πιο πάνω πίνακες απεικονίζουν μια παραδειγματική καταγραφή στοιχείων σε ένα λεξικό δεδομένων, ώστε να δοθεί μία πρακτική εικόνα για τον τρόπο σχεδιασμού ενός τέτοιου λεξικού.

5.2.9. Διαγράμματα HIPO *(Hierarchy and Input-Process-Output)*

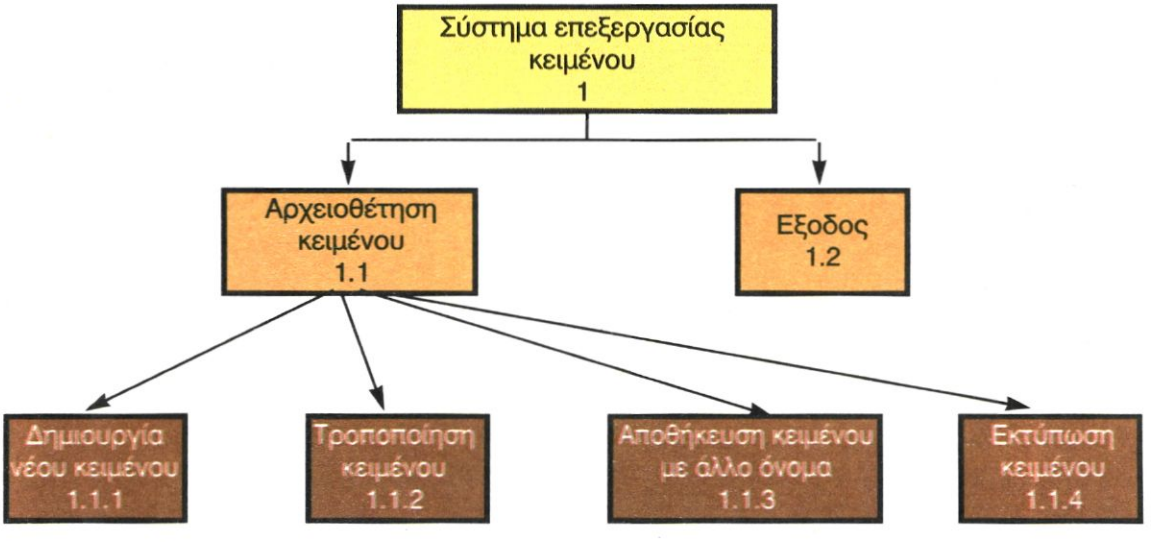
Μια άλλη κατηγορία διαγραμμάτων, αρκετά δημοφιλής στους σύγχρονους αναλυτές συστημάτων, είναι τα διαγράμματα "HIPO", δηλαδή "ιεραρχικά διαγράμματα εισροών - επεξεργασίας - εκροών". Για πολλούς αναλυτές αποτελούν μια εναλλακτική προσέγγιση στα ΔΡΔ, ενώ για άλλους ένα επιπλέον εργαλείο. Τα διαγράμματα HIPO παρουσιάζουν με ιεραρχικό τρόπο τις επιμέρους λειτουργίες (functions), που συνθέτουν ένα σύστημα. Σε ένα σχετικά πολύπλοκο σύστημα, ένα διάγραμμα HIPO μπορεί να εκτείνεται σε πολλές σελίδες. Τα σύμβολα που χρησιμοποιούνται σε διαγράμματα αυτού του τύπου είναι μόνο δύο:

Ένα βέλος → και ένα κλειστό ορθογώνιο

Η λογική ενός τέτοιου διαγράμματος είναι να παρουσιάσει με ιεραρχικό τρόπο και σε πολλαπλά επίπεδα τις λειτουργίες του συστήματος, έτσι ώστε όλες οι υπολειτουργίες μιας ορισμένης λειτουργίας να βρίσκονται σε ένα ιεραρχικό επίπεδο πιο κάτω από αυτήν. Κάθε λειτουργία συμβολίζεται με το κλειστό ορθογώνιο, ενώ το βέλος συνδέει από πάνω προς τα κάτω κάθε λειτουργία με τις υπολειτουργίες της.

Το παρακάτω **σχήμα 5.6** περιγράφει τις βασικές λειτουργίες ενός προγράμματος επεξεργασίας κειμένου, με τη χρήση ενός διαγράμματος HIPO.

Ένα σωστά σχεδιασμένο διάγραμμα HIPO, μπορεί να προσφέρει μια τόσο σαφή εικόνα για την εφαρμογή, όσο δεν θα μπορούσαν πολλές δεκάδες λέξεις να προσφέρουν.



Σχήμα 5.6. – HIPO διάγραμμα, που απεικονίζει τις βασικές λειτουργίες ενός στοιχειώδους επεξεργαστή κειμένου.

Πέρα από τα διαγράμματα που ως τώρα παρουσιάστηκαν, υπάρχουν και άλλα, λιγότερο σημαντικά και σπανιότερα χρησιμοποιούμενα, των οποίων, όμως, η παρουσίαση είναι πέρα από το πλαίσιο και τους στόχους του βιβλίου αυτού. Περισσότερα μπορεί να βρει κανείς ανατρέχοντας στη σχετική βιβλιογραφία στο τέλος του κεφαλαίου.

5.2.10. Αρχές, διαδικασίες και λήψη αποφάσεων κατά την ανάπτυξη του συστήματος

Κάθε καλά σχεδιασμένο σύστημα βασίζεται στην πιστή εφαρμογή μιας σειράς αρχών, οι οποίες διαμορφώνονται μέσα από την επαφή με τον πελάτη και προσδιορίζουν τους στόχους του συστήματος. Οι αρχές αυτές είναι ένα σύνολο κανόνων που πρέπει να εφαρμοστούν, ώστε το σύστημα να ανταποκρίνεται στις προσδοκίες, που ο πελάτης έχει από αυτό. Οι κανόνες αυτοί, εφόσον βέβαια είναι προσδιορισμένοι με σαφήνεια και ακρίβεια, μπορούν σχετικά εύκολα να μετατραπούν σε αλγόριθμους, που θα αποτελέσουν τη βάση ανάπτυξης των προγραμμάτων, τα οποία θα υποστηρίζουν τις λειτουργίες του συστήματος.

Τα εργαλεία που συνήθως χρησιμοποιούνται για ακριβή περιγραφή των αρχών του συστήματος, είναι οι **πίνακες αποφάσεων** και το **δομημένο κείμενο.**

Η παρουσίαση τέτοιων περισσότερο εξειδικευμένων εργαλείων ξεφεύγει από τα πλαίσια του βιβλίου αυτού. Για εκείνους πάντως, που επιθυμούν μια πιο ολοκληρωμένη πληροφόρηση πάνω στα εργαλεία αυτά, παρατίθενται πιο κάτω σχετικοί ορισμοί και παραδείγματα.

Το δομημένο κείμενο *(structured text)* αποτελείται από προτάσεις, που συντάσσονται στην ομιλούμενη γλώσσα και έχουν σαν στόχο να περιγράψουν τις διαδικασίες που θα υποστηρίζει το σύστημα, με τρόπο κατανοητό, όχι μόνο από τους προγραμματιστές, αλλά και από τους υπεύθυνους, που έχει ορίσει ο πελάτης για την παρακολούθηση της πορείας ανάπτυξης του συστήματος.

Μερικοί βασικοί κανόνες σύνταξης τέτοιων κειμένων είναι οι εξής:

- Πρέπει να χρησιμοποιούνται μόνο προστακτικά ρήματα.
- Οι όροι και τα χρησιμοποιούμενα ουσιαστικά πρέπει να υπάρχουν στο λεξικό δεδομένων.
- Δεν πρέπει να χρησιμοποιούνται υποσημειώσεις.
- Τα επίθετα και τα επιρρήματα που χρησιμοποιούνται πρέπει να είναι καλά ορισμένα.
- Οι λογικές κατασκευές που χρησιμοποιούνται πρέπει να είναι οι ελάχιστες δυνατές (επανάληψη - απόφαση - ακολουθία).

Τυπική μορφή ενός δομημένου κειμένου δίνεται στο επόμενο παράδειγμα στον **πίνακα 5.6.**

Για κάθε ΠΑΡΑΓΓΕΛΙΑ_ΠΕΛΑΤΗ εκτέλεσε τα πιο κάτω βήματα:
Αν ΠΡΟΚΑΤΑΒΟΛΗ μικρότερη από ΣΥΝΟΛΙΚΗ_ΑΞΙΑ_ΠΑΡΑΓΓΕΛΙΑΣ τότε:
Βρες στο ΑΡΧΕΙΟ_ΠΕΛΑΤΩΝ το ΛΟΓΑΡΙΑΣΜΟΣ_ΠΕΛΑΤΗ με κλειδί αναζήτησης τον ΑΡΙΘΜΟ_ΛΟΓΑΡΙΑΣΜΟΥ_ΠΕΛΑΤΗ.

Αν ΛΟΓΑΡΙΑΣΜΟΣ_ΠΕΛΑΤΗ ανύπαρκτος τότε:
α) τοποθέτησε 0 στο ΙΣΤΟΡΙΚΟ_ΠΙΣΤΩΣΗΣ.
β) τοποθέτησε 0 στο ΥΨΟΣ_ΕΡΓΑΣΙΩΝ.
γ) τοποθέτησε 500000 στο ΟΡΙΟ_ΠΙΣΤΩΣΗΣ.
δ) τοποθέτησε 0 στο ΤΡΕΧΟΝ_ΥΠΟΛΟΙΠΟ.
ε) καταχώρησε τα ΣΤΟΙΧΕΙΑ_ΠΙΣΤΩΣΗΣ στο ΑΡΧΕΙΟ_ΠΕΛΑΤΩΝ.

Υπολόγισε το ΠΡΟΤΕΙΝΟΜΕΝΟ_ΙΣΟΖΥΓΙΟ με βάση τον τύπο:
ΠΡΟΤΕΙΝΟΜΕΝΟ_ΙΣΟΖΥΓΙΟ = ΤΡΕΧΟΝ_ΥΠΟΛΟΙΠΟ + ΣΥΝΟΛΙΚΗ_ΑΞΙΑ_ΠΑΡΑΓΓΕΛΙΑΣ - ΠΡΟΚΑΤΑΒΟΛΗ.

Επίλεξε την ΕΝΕΡΓΕΙΑ_ΠΙΣΤΩΣΗΣ χρησιμοποιώντας τα ΣΤΟΙΧΕΙΑ_ΠΙΣΤΩΣΗΣ και τον πίνακα αποφάσεων ΠΙΣΤΩΣΗ.

Αν η ενέργεια είναι "ΕΓΚΡΙΣΗ ΠΙΣΤΩΣΗΣ" τότε κατάγραψε την ΕΓΚΕΚΡΙΜΕΝΗ_ΠΑΡΑΓΓΕΛΙΑ αλλιώς κατάγραψε την ΑΠΟΡΡΙΦΘΕΙΣΑ_ΠΑΡΑΓΓΕΛΙΑ.

Πίνακας 5.6. – Παράδειγμα δομημένου κειμένου.

Τα δομημένα κείμενα βοηθούν αποφασιστικά στα επόμενα βήματα της διαδικασίας ανάπτυξης του συστήματος και ιδίως στη δημιουργία του κώδικα των προγραμμάτων. Χρησιμοποιούνται επίσης από τους αναλυτές για την περιγραφή των σύνθετων διαδικασιών του συστήματος.

Οι πίνακες απόφασης *(decision tables)* αποτελούν σύνηθες εργαλείο για την περιγραφή και την αποτύπωση των κανόνων και των συνθηκών, κάτω από τις οποίες λαμβάνονται διάφορες αποφάσεις, στο πλαίσιο της λειτουργίας του συστήματος. Ο **πίνακας 5.7.** αποτελεί ένα παράδειγμα πίνακα απόφασης, όπου με βάση κάποιους κανόνες (κανόνες 1,2,3 και 4), λαμβάνονται οι αποφάσεις που ορίζονται στην κάτω αριστερή πλευρά του, σε συνδυασμό με τις συνθήκες που αναφέρονται σαν συνθήκες 1,2,3,4 και 5 και αποτελούν τις γραμμές του πίνακα. Στον πίνακα αυτό διακρίνονται μόνο δύο καταστάσεις ανά συνθήκη και κανόνα. Αν ισχύει η συνθήκη, τότε η κατάσταση χαρακτηρίζεται ως "Α" (αληθής). Αν όχι, τότε η κατάσταση χαρακτηρίζεται ως "Ψ" (ψευδής).

	ΚΑΝΟΝΑΣ 1	ΚΑΝΟΝΑΣ 2	ΚΑΝΟΝΑΣ 3	ΚΑΝΟΝΑΣ 4
Συνθήκη 1	Α	Ψ	Ψ	Ψ
Συνθήκη 2	Α	Α	Ψ	Ψ
Συνθήκη 3	Α	Α	Ψ	Ψ
Συνθήκη 4	Α	Α	Ψ	Ψ
Συνθήκη 5	Α	Α	Α	Ψ
Απόφαση 1	Χ	Χ		
Απόφαση 2			Χ	Χ

Πίνακας 5.7. - Παράδειγμα πίνακα απόφασης.

Οι πίνακες απόφασης αν και χρησιμοποιούνται από αρκετούς αναλυτές, αποτελούν εξειδικευμένο εργαλείο, οι λεπτομέρειες για το οποίο ξεφεύγουν από τους σκοπούς του βιβλίου αυτού.

Ανακεφαλαίωση / περίληψη

Οι απαιτήσεις εκφράζουν τις προσδοκίες των μελλοντικών χρηστών από το νέο σύστημα, προσδιορίζοντας τι ανάγκες πρέπει να ικανοποιηθούν και όχι πώς θα γίνει αυτό.

Οι απαιτήσεις διακρίνονται σε λειτουργικές, οι οποίες προσδιορίζουν τι είδους επαφές πρέπει να υπάρχουν ανάμεσα στο σύστημα και το περιβάλλον του και μη λειτουργικές, οι οποίες αποτελούν το σύνολο όλων των περιορισμών που θέτει η πραγματικότητα, μειώνοντας τις υπάρχουσες επιλογές για την ανάπτυξη του συστήματος.

Οι κάθε είδους απαιτήσεις πρέπει να αναλύονται με πληρότητα, σαφήνεια και σε ρεαλιστική βάση, ώστε να λαμβάνουν υπόψη τους διαθέσιμους πόρους και τους περιορισμούς, που τίθενται από την πραγματικότητα.

Τα συνηθέστερα εργαλεία που χρησιμοποιούνται, είτε για την αποτύπωση του ήδη υπάρχοντος συστήματος, αν υπάρχει τέτοιο, είτε των απαιτήσεων από το νέο σύστημα, είναι τα διαγράμματα ροής δεδομένων, τα διαγράμματα "HIPO", οι πίνακες απόφασης και διάφορες άλλες τεχνικές αποτύπωσης.

Η ανάλυση από πολλούς θεωρείται το σημαντικότερο από τα στάδια του λεγόμενου κύκλου ζωής του συστήματος, (βλ. κεφ.7 και κεφ. 10). Ο λόγος είναι ότι μια σωστά δομημένη και πλήρης ανάλυση προσφέρει το ιδανικό έδαφος για ένα επιτυχημένο και σύντομο τελικό σχεδιασμό του συστήματος.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Τι εννοεί κανείς αναφερόμενος σε απαιτήσεις χρηστών από ένα σύστημα; Τι σημαίνει ανάλυση απαιτήσεων; Να δοθούν συνοπτικοί ορισμοί.
2. Πιστεύετε ότι είναι σημαντικό να διευκρινίζονται πλήρως οι απαιτήσεις των χρηστών από ένα σύστημα και αν ναι γιατί; Να δοθεί σύντομη απάντηση.
3. Να επιλεγούν με τη συμπλήρωση του αντίστοιχου τετραγωνίδιου με το σύμβολο Χ, όσες από τις παρακάτω λέξεις ή φράσεις πιστεύετε ότι αντιστοιχούν σε κριτήρια διαμόρφωσης απαιτήσεων:

Πληρότητα	❑
Ακρίβεια στον καθορισμό	❑
Σχολαστικότητα	❑
Αναγνωσιμότητα	❑
Επαληθευσιμότητα	❑

4. Να συνδέσετε με μία γραμμή, τα πιο κάτω σύμβολα που χρησιμοποιούνται σε ένα ΔΡΔ, με τη σημασία που αντιστοιχεί στο κάθε ένα από αυτά:

Ροή	•	•
Διεργασία	•	•
Αποθήκη δεδομένων	•	•
Εξωτερική οντότητα	•	•

5. Να σχεδιάστε σε ένα φύλο χαρτί, με ή χωρίς τη βοήθεια Η/Υ, ένα ερωτηματολόγιο, το οποίο προορίζεται για συμπλήρωση από τον ιδιοκτήτη της βιντεολέσχης "ΤΑΙΝΙΑ Α.Ε.", που χρησιμοποιήθηκε σαν παράδειγμα σε πολλές περιπτώσεις στο βιβλίο. Το ερωτηματολόγιο πρέπει να συνταχθεί με τέτοιο τρόπο, ώστε να παρέχει πληροφορίες, αποκλειστικά και μόνο για το πώς γίνεται μέχρι τη στιγμή εκείνη η διαχείριση της παραγγελίας ενός πελάτη, που ζητά να νοικιάσει μια συγκεκριμένη ταινία. Να υποθέσετε, ότι μέχρι στιγμής ο πελάτης δεν έχει αυτοματοποιήσει την πιο πάνω διαδικασία και τη διεκπεραιώνει χειροκίνητα.
6. Να ανταλλαγούν απόψεις μεταξύ των μαθητών σχετικά με τα διάφορα στάδια της ανάλυσης και κατά πόσο πιστεύουν ότι αυτά πρέπει να ακολουθούνται κατά γράμμα σε όλων των ειδών τα υπό ανάπτυξη συστήματα. Υπάρχει δυνατότητα ανάλογα με την εφαρμογή να παραλειφθούν ή να συμπυκνωθούν κάποια από αυτά; Θεωρείτε ότι πρέπει πάντα να αποτελούν διαφορετικά άτομα την ομάδα ανάλυσης, σε σχέση με τις άλλες ομάδες που αναλαμβάνουν τα επιμέρους στάδια ανάπτυξης του συστήματος; Πότε πιστεύετε ότι είναι δυνατό ένα και μόνο άτομο να αναλάβει ολόκληρο τον κύκλο ζωής ενός συστήματος;

Γλωσσάριο - Ευρετήριο όρων

διαγράμματα HIPO *(Hierarchy and Input - Process - Output)* - Ειδική κατηγορία διαγραμμάτων που χρησιμοποιούνται στη φάση της ανάλυσης ενός συστήματος για να παρουσιάσουν τις βασικές λειτουργίες ενός συστήματος και τον τρόπο σύνδεσής τους σε ιεραρχικά δένδρα [107].

διαγράμματα ροής δεδομένων *(data flow diagrams)* - Τα διαγράμματα αυτά γνωστά και από τη συντομογραφία τους σαν ΔΡΔ, απεικονίζουν με ειδικά σύμβολα, τη ροή των δεδομένων μεταξύ των διαδικασιών ενός συστήματος, υποβοηθώντας σε μεγάλο βαθμό την κατανόηση των εννοιών που εμπλέκονται στο σύστημα [100].

λειτουργικές απαιτήσεις - Περιγράφουν τι είδους επαφή πρέπει να υπάρχει ανάμεσα στο σύστημα και το περιβάλλον του. Δηλαδή, πώς το σύστημα πρέπει να αποκρίνεται όταν δέχεται συγκεκριμένα δεδομένα και τι είδους αποτελέσματα πρέπει να παράγει ανάλογα με τα εισερχόμενα αιτήματα [96].

λεξικό δεδομένων *(data dictionary)* - Ευρετήριο στο οποίο επεξηγούνται όλοι οι όροι που χρησιμοποιούνται στη φάση της ανάλυσης και του σχεδιασμού και αναφέρονται σε διεργασίες, ροές, αποθήκες δεδομένων αλλά και στα ίδια τα δεδομένα [106].

Βιβλιογραφία

Hawryszkiewycz I.T., "Introduction to Systems Analysis and Design", Prentice Hall, 1988

Pressman R.S., "Software Engineering", McGraw Hill, 1992.

Yourdon Ed., "Modern Structured Analysis", Prentice-Hall, 1988.

Βατικιώτης Αντ., "Ανάλυση – σχεδίαση συστημάτων επεξεργασίας στοιχείων", Α.Βατικιώτης, 1989

Λαοπόδης Β., "Ανάλυση και σχεδιασμός συστημάτων", Νέες Τεχνολογίες, 1996.

ΚΕΦΑΛΑΙΟ 6

ΟΡΓΑΝΩΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

ΠΕΡΙΕΧΟΜΕΝΑ

6.1. Διάκριση και τύποι δεδομένων
6.2. Κανονικοποίηση και βάσεις δεδομένων
6.3. Πίνακες και συσχετίσεις πινάκων

ΕΙΣΑΓΩΓΗ

Η αξιόπιστη και ασφαλής οργάνωση των δεδομένων είναι από τους ακρογωνιαίους λίθους της δημιουργίας κάθε αξιόλογου συστήματος. Ο παραδοσιακός τρόπος οργάνωσης σε αρχεία έχει τα πλεονεκτήματά του, στην περίπτωση που δεν απαιτείται πολύπλοκη οργάνωση μεγάλου όγκου σχετιζόμενων μεταξύ τους δεδομένων.

- Είναι, όμως, ο τρόπος αυτός αποτελεσματικός και ενδεδειγμένος στην περίπτωση που μια εφαρμογή έχει σαν κύριο στόχο την ευέλικτη διαχείριση μεγάλου όγκου δεδομένων;
- Πέρα από την αποτελεσματική διαχείριση των δεδομένων, πώς αντιμετωπίζεται το ζήτημα της ασφάλειάς τους;
- Πόσο σημαντική είναι η συχνή αναπροσαρμογή της οργάνωσης των δεδομένων και πόσο το κόστος μιας τέτοιας αναπροσαρμογής είναι εύκολο να αντιμετωπιστεί;

Η απάντηση που όλο και πιο συχνά δίνεται στα σύγχρονα περιβάλλοντα εργασίας για την αντιμετώπιση των πιο πάνω προβληματισμών, είναι η επιλογή ενός συστήματος διαχείρισης βάσεων δεδομένων, όπως έχει ήδη εξηγηθεί στο κεφάλαιο 1.

Τα πιο διαδεδομένα από αυτά τα συστήματα βασίζονται στο λεγόμενο σχεσιακό πρότυπο *(relational model)*, όπως αυτό παρουσιάστηκε από τον Edgar Codd στις αρχές της δεκαετίας του 70 και παρουσιάστηκε επίσης στο κεφάλαιο 1.

Σύμφωνα με το πρότυπο αυτό, τα δεδομένα οργανώνονται σε πίνακες, οι οποίοι είναι δομές που μοιάζουν με αρχεία. Όμως, οι δομές αυτές δεν αποθηκεύονται ως αυτόνομες αναγνωρίσιμες οντότητες, όπως τα αρχεία.

Η αποθήκευση των δεδομένων τους γίνεται με τρόπο μη αντιληπτό από το χρήστη του συστήματος και με τη βοήθεια του μηχανισμού του σχεσιακού συστήματος βάσεων δεδομένων. Ο μηχανισμός αυτός διασφαλίζει τη συσχέτιση των πινάκων, έτσι ώστε να είναι δυνατή η λήψη δεδομένων από περισσότερους από ένα πίνακες, με σχετικά απλό τρόπο, χωρίς αναγκαστικά πολύπλοκο προγραμματισμό.

Τα συστήματα αυτά αναμένεται να εξελιχθούν ακόμη περισσότερο τα επόμενα χρόνια, προσφέροντας δυνατότητες, που πριν λίγα χρόνια θα χαρακτηρίζονταν εξωπραγματικές.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να αναφέρει τις πιο διαδεδομένες τεχνικές που χρησιμοποιούνται για την αποθήκευση των δεδομένων σε αρχεία.
2. Να παρουσιάζει ορισμένες πρόσθετες γνώσεις που αφορούν στα συστήματα βάσεων δεδομένων και ιδιαίτερα των σχεσιακών.
3. Να μπορεί να φέρει τα δεδομένα των σχεσιακών βάσεων μέχρι και την τρίτη κανονική μορφή, αφού πρώτα διακρίνει τις οντότητες που συνθέτονται από τα δεδομένα ενός προβλήματος.
4. Να διακρίνει τις έννοιες που σχετίζονται με τους πίνακες και τους μηχανισμούς διασφάλισης της ακεραιότητας των δεδομένων της βάσης.

6.1. ΔΙΑΚΡΙΣΗ ΚΑΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Τα δεδομένα που ένα αυτοματοποιημένο σύστημα διαχειρίζεται και από τα οποία συνθέτει τις πληροφορίες που παρέχει προς τα έξω, είναι η καρδιά του ίδιου του συστήματος. Σύστημα χωρίς αποθηκευμένες πληροφορίες είναι στην ουσία ένα άχρηστο σύστημα.

Τα δεδομένα που χρησιμοποιεί το σύστημα δεν είναι δυνατό να βρίσκονται όλα, την ίδια στιγμή, στην κεντρική μνήμη του υπολογιστή. Αποθηκεύονται σε μαγνητικά μέσα αποθήκευσης (σκληρούς δίσκους, ταινίες, δισκέτες) ή οπτικά μέσα αποθήκευσης (CD μιας ή πολλών εγγραφών) και ανακαλούνται όποτε αυτό απαιτηθεί. Για να μπορεί να γίνει σωστή αξιοποίηση αυτών των δεδομένων από τις εφαρμογές που τα χρησιμοποιούν, πρέπει να έχει επιλεγεί ο καλύτερος δυνατός τρόπος οργάνωσής τους στα μέσα που είναι αποθηκευμένα, ώστε να είναι εφικτή η αποτελεσματική και γρήγορη ανάκτησή τους.

6.1.1. Τρόποι οργάνωσης των δεδομένων.

Ο τρόπος με τον οποίο τα δεδομένα που χρησιμοποιούνται από ηλεκτρονικούς (ψηφιακούς) υπολογιστές, οργανώνονται στα μέσα αποθήκευσής τους, εξαρτάται από το πώς προορίζονται να αξιοποιηθούν.

Παράδειγμα:

Ο πιο παραδοσιακός και περισσότερο διαδεδομένος τρόπος οργάνωσης είναι σε αρχεία εγγραφών. Ας φανταστούμε ένα σχολείο όπου οι καθηγητές καταχωρούν σε καρτέλες τα στοιχεία των μαθητών, έτσι ώστε κάθε καρτέλα να αντιστοιχεί σε διαφορετικό μαθητή. Τα στοιχεία αυτά είναι:

- Ονοματεπώνυμο μαθητή.
- Όνομα πατέρα.
- Όνομα μητέρας.
- Ημερομηνία γέννησης.
- Τμήμα στο οποίο είναι ενταγμένος.
- Αριθμός απουσιών.
- Βαθμολογίες (δεν αναλύονται εδώ οι επιμέρους λεπτομέρειες της οργάνωσης των σχετικών στοιχείων).

Ας υποτεθεί, τέλος, για χάρη του παραδείγματος, ότι όλες οι καρτέλες φυλάσσονται σε ένα συγκεκριμένο συρτάρι και ας προσπαθήσουμε να φανταστούμε τη διαδικασία αυτή στο μηχανογραφημένο αντίστοιχό της.

- Το αντίστοιχο του συρταριού είναι το **αρχείο** *(file)*
- Κάθε καρτέλα μαθητή αποτελεί μια **εγγραφή** *(record)* του αρχείου.
- Τέλος, κάθε στοιχείο που περιλαμβάνεται στην καρτέλα είναι ένα **πεδίο** *(field)*.

Ορισμός

Πεδίο, λοιπόν, είναι ένα από τα επιμέρους στοιχεία-πληροφορίες που συνθέτουν την εγγραφή, όπως αυτή θα καταχωρηθεί σε ένα αρχείο. Το πεδίο, καθώς γίνεται η επεξεργασία του από τον Η/Υ, δεσμεύει κάποιο χώρο στην κεντρική ή στη δευτερεύουσα μνήμη του υπολογιστή, χώρο ο οποίος μετριέται σε bytes. Ο αριθμός των απαιτούμενων bytes, είναι το **μήκος του πεδίου** *(field length)*.

Ανάλογα με το περιεχόμενο τους τα πεδία μπορούμε να τα διακρίνουμε σε διάφορες κατηγορίες. Οι κυριότερες από αυτές είναι:

- **Αλφαριθμητικά** *(alphanumeric)*, όταν περιέχουν αλφαβητικούς χαρακτήρες, ειδικούς χαρακτήρες, αριθμούς ή συνδυασμούς αυτών.
- **Αριθμητικά** *(numeric)*, όταν περιέχουν μόνον αριθμούς.
- **Αλφαβητικά** *(alphabetic)*, όταν περιέχουν μόνον αλφαβητικούς χαρακτήρες.
- **Ημερομηνίας** *(date)*, όταν έχουν τη δυνατότητα αποθήκευσης ημερομηνιών με διάφορες μορφές (π.χ., 12-12-98 ή 12-03-1998 κ.ο.κ., ανάλογα με το προγραμματιστικό περιβάλλον που χρησιμοποιείται).
- **Δυαδικά** *(binary)*, όταν είναι κατάλληλα να αποθηκεύουν ειδικού τύπου δεδομένα, όπως εικόνα, ήχο, video κ.λ.π.).
- **Λογικά** *(logical)*, όπου επιτρέπονται μόνο δύο τιμές που αντιστοιχούν σε δύο διακριτές καταστάσεις: Αλήθεια *(True)* ή Ψεύδος *(False)*.
- **Σημειώσεων** *(memo)*, όπου είναι επιτρεπτή η εισαγωγή κειμένου με μεταβλητό μήκος, το οποίο συνήθως χρησιμοποιείται για την αποθήκευση σημειώσεων (κειμένου), που μπορεί, ανάλογα και με το περιβάλλον ανάπτυξης, να είναι και αρκετά μεγάλου μεγέθους.

Ιδανική λύση θα ήταν βέβαια τα μαγνητικά μέσα αποθήκευσης δεδομένων, όπως οι σκληροί δίσκοι, να χαρακτηρίζονταν από ταχύτητες προσπέλασης, παραπλήσιες με εκείνες των κεντρικών μνημών και να μπορούσαν οι επεξεργαστές, να τους αξιοποιήσουν απευθείας. Τότε θα μπορούσαν να επιτευχθούν εκπληκτικές για τα σημερινά δεδομένα ταχύτητες επεξεργασίας.

Πολλά πεδία που ανήκουν στην ίδια λογική ενότητα αποτελούν μια **εγγραφή** *(record)*. **Μήκος εγγραφής** *(record length)* είναι το άθροισμα του μήκους των πεδίων που αποτελούν την εγγραφή. **Δομή εγγραφής** (ή γραμμογράφηση) *(record layout)* είναι ο τρόπος με τον οποίο οργανώνονται τα πεδία ώστε να σχηματίσουν την εγγραφή. Το βασικό στοιχείο που καθορίζει τη δομή εγγραφής είναι η σειρά με την οποία εμφανίζονται τα πεδία μέσα στην εγγραφή, δηλαδή ποιο είναι πρώτο, ποιο δεύτερο κ.ο.κ.

Ορισμός

Αρχείο *(file)* είναι ένα σύνολο από εγγραφές του ίδιου τύπου που είναι καταχωρημένες σε ένα (ή και περισσότερα, σύμφωνα με τις δυνατότητες της σημερινής τεχνολογίας) μέσο αποθήκευσης. Τα προγράμματα εφαρμογών διαβάζουν ή γράφουν στα αρχεία δεδομένα με τη μορφή εγγραφών.

Διάβασμα *(read)* από αρχείο σημαίνει στη γλώσσα των υπολογιστών ότι το πρόγραμμα πηγαίνει στο μέσο αποθήκευσης του αρχείου, παίρνει δεδομένα και τα φέρνει στην κεντρική μνήμη του υπολογιστή για να τα επεξεργαστεί σύμφωνα με το σχεδιασμό του. **Γράψιμο** *(write)* σημαίνει ότι, δεδομένα που βρίσκονται στην κεντρική μνήμη του υπολογιστή επεξεργασμένα ή όχι από το πρόγραμμα, αποθηκεύονται στο αρχείο.

Με τις διαδικασίες γραψίματος και διαβάσματος σε ένα αρχείο τα προγράμματα εκμεταλλεύονται, τόσο τη μεγάλη αποθηκευτική ικανότητα των μαγνητικών μέσων αποθήκευσης του συστήματος, όσο και τη μεγάλη ταχύτητα της κεντρικής μνήμης του, μεταφέροντας από τον ένα χώρο στον άλλο, το μέρος του αρχείου που πρόκειται να επεξεργαστούν. Η αναγκαιότητα αυτής της τεχνικής φαίνεται στο πιο κάτω παράδειγμα:

Παράδειγμα:

Ας υποτεθεί ότι κάποιο πρόγραμμα χρειάζεται δεδομένα, αποθηκευμένα σε ένα αρχείο 1.000.000 εγγραφών. Προκειμένου να εντοπίσει αυτό ακριβώς που χρειάζεται μέσα από ένα τέτοιο πλήθος εγγραφών, ή θα έπρεπε να φέρει όλες τις εγγραφές στην κεντρική μνήμη, ή με κάποιο αλγόριθμο επιλογής, να αναζητήσει και να ανασύρει από το αρχείο μόνο τις επιθυμητές εγγραφές. Στην πρώτη περίπτωση το πιθανότερο είναι ότι δεν θα επαρκούσε η διαθέσιμη κεντρική μνήμη, την οποία, άλλωστε, μπορεί να χρησιμοποιούν ταυτόχρονα και άλλα προγράμματα που εκτελούνται ("τρέχουν") στον υπολογιστή. Συνεπώς απομένει η δεύτερη λύση, δηλαδή να αναζητήσει και να ανασύρει από το αρχείο μόνο τις επιθυμητές εγγραφές. Για να γίνει όμως αυτό θα πρέπει να υπάρχει η κατάλληλη οργάνωση των δεδομένων μέσα στα αρχεία. Το πως οργανώνονται τα δεδομένα μέσα στα αρχεία περιγράφεται στο επόμενο κεφάλαιο.

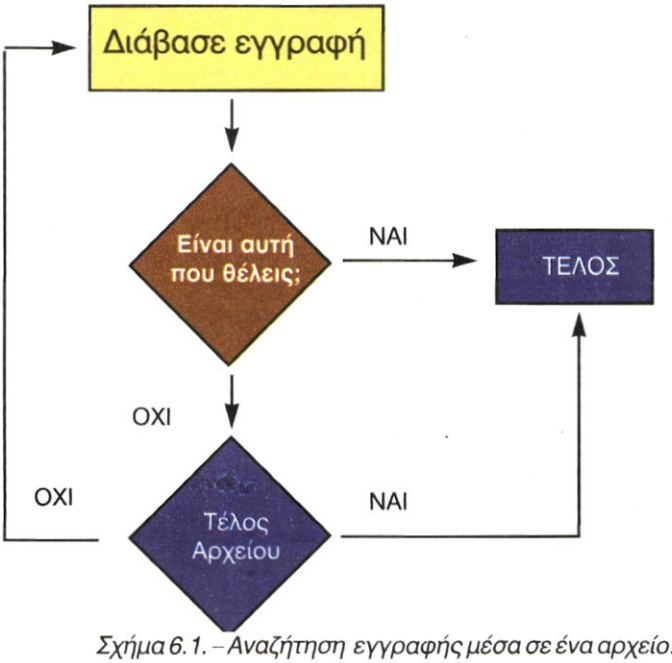
6.1.2. Τρόποι οργάνωσης των δεδομένων μέσα στα αρχεία

Τα αρχεία, ανάλογα με τον τρόπο οργάνωσης των δεδομένων μέσα σ' αυτά, διακρίνονται σε τρεις βασικές κατηγορίες: τα σειριακά *(sequential)*, τα αρχεία με δείκτες (ή αρχεία δεικτών ή δεικτοδοτημένα) *(indexed)* και τα τυχαίας προσπέλασης *(random access)*. Για την κατανόηση των διαφορών στην οργάνωση των παραπάνω αρχείων θα δοθούν τρία παραδείγματα:

Παράδειγμα:

Ας υποτεθεί ότι ο διευθυντής ενός σχολείου που χρησιμοποιεί κάποια εφαρμογή λογισμικού για τη διαχείριση των στοιχείων των μαθητών του, επιθυμεί να δει στην οθόνη του υπολογιστή τα στοιχεία του μαθητή "ΨΩΜΙΑΔΗ ΝΙΚΟΛΑΟΥ". Ας υποτεθεί ακόμη ότι τα στοιχεία αυτά είναι αποθηκευμένα σε αρχείο με τύπο εγγραφής αντίστοιχο του παραδείγματος με τις καρτέλες που αναφέρθηκε πιο πάνω.

Τα στοιχεία των μαθητών εισάγονται συνήθως στο σύστημα με πληκτρολόγηση, με τη βοήθεια κάποιας φόρμας εισαγωγής (μάσκας οθόνης), που παρέχει η εφαρμογή. Αν, όμως, κάθε νέα εγγραφή που εισάγεται στο σύστημα, προστίθεται στο τέλος κάποιου αρχείου, τότε δημιουργείται το εξής πρόβλημα: Αν σε κάποια μεταγενέστερη χρονική στιγμή, ο διευθυντής ζητήσει από το πρόγραμμα τα στοιχεία του συγκεκριμένου μαθητή, το πρόγραμμα ακολουθεί τα βήματα που περιγράφονται στο διάγραμμα του σχήματος 6.1 (ξεκινώντας από την πρώτη εγγραφή του αρχείου):



Σχήμα 6.1. – Αναζήτηση εγγραφής μέσα σε ένα αρχείο.

Στην περίπτωση που η αναζητούμενη εγγραφή είναι, για παράδειγμα, η εκατοστή, ένα σύγχρονο σύστημα θα την ανασύρει στην κεντρική μνήμη του Η/Υ για επεξεργασία (εμφάνιση), εξαιρετικά γρήγορα. Αν όμως είναι η εκατομμυριοστή, τότε ο χρόνος απόκρισης του προγράμματος θα είναι αυξημένος. Το πόσο αυξημένος θα είναι εξαρτάται από πολλούς παράγοντες και ιδιαίτερα από τον εξοπλισμό του υπολογιστή (κυρίως την ταχύτητα αναζήτησης στο αποθηκευτικό μέσο που βρίσκεται το αρχείο).

Κατά την αναζήτηση μιας εγγραφής με τον παραπάνω τρόπο, η κεφαλή ανάγνωσης του μαγνητικού μέσου κινείται σειριακά, δηλαδή μετακινείται από τη μια εγγραφή στην επόμενη, την ανασύρει στην κεντρική μνήμη, γίνεται ο έλεγχος από το πρόγραμμα αν η εγγραφή αυτή είναι η επιθυμητή και αν όχι, τότε πηγαίνει στην αμέσως επόμενη στη σειρά.

Εξαιτίας της λογικής με την οποία γίνεται, τόσο η αποθήκευση, όσο και η αναζήτηση των εγγραφών στο παράδειγμα αυτό, η συγκεκριμένη οργάνωση του αρχείου ονομάζεται σειριακή και το αρχείο χαρακτηρίζεται σαν σειριακό (sequential).

Παράδειγμα: Ας υποθέσουμε ότι αναζητούμε το τηλέφωνο του φίλου μας Νίκου Ψωμιάδη στον τηλεφωνικό κατάλογο. Προφανώς δεν ανοίγουμε τον κατάλογο αρχίζοντας να ψάχνουμε από την πρώτη σελίδα με τη σειρά όλα τα ονόματα, γιατί έτσι θα σπαταλούσαμε άσκοπα την ώρα μας. Εκείνο που κάθε λογικός άνθρωπος θα σκεφτεί, είναι να ψάξει

από πού ξεκινά το γράμμα Ψ, μετά να αναζητήσει το συνδυασμό του Ψ με το ω και συνεχίζοντας με τη λογική αυτή, να εντοπίσει το όνομα που ψάχνει. Άρα, στην περίπτωση αυτή, που τα δεδομένα (τα ονόματα στον κατάλογο) είναι τόσο πολλά, η απλή αναζήτησή τους δεν είναι αποτελεσματική.

Προτιμήθηκε, λοιπόν, μια άλλου είδους οργάνωση, όπου στο εξώφυλλο κάθε τόμου δηλώνεται σαν δείκτης του εύρους των ονομάτων που περιέχει ο συγκεκριμένος τόμος, π.χ., αναγράφονται τα αρχικά Α-Κ και εννοείται ότι περιέχονται τα επίθετα που ξεκινούν από Α μέχρι και αυτά που ξεκινούν από Κ. Παράλληλα, μέσα σε κάθε τόμο υπάρχουν και εκεί δείκτες, οι οποίοι οδηγούν στα αρχικά των επιθέτων που ταυτίζονται με τα αναζητούμενα. Παρόμοια λογική (ύπαρξη δεικτών ή πινάκων περιεχομένων) εφαρμόζεται στις εγκυκλοπαίδειες, στα περιεχόμενα των βιβλίων κάθε είδους, σε τιμοκαταλόγους και σε πολλές άλλες περιπτώσεις της καθημερινής ζωής.

Αν προσπαθήσει να υλοποιήσει κάποιος ένα από τα πιο πάνω παραδείγματα μηχανογραφικά, τότε μια από τις καλύτερες επιλογές που έχει στη διάθεσή του είναι να οργανώσει τα αρχεία με δείκτες *(index files)*.

Η μέθοδος αυτή είναι η καταλληλότερη, όταν οι πληροφορίες μπορούν να οργανωθούν σε ένα μόνο αρχείο και οι αναζητήσεις που συνήθως κάνουμε, βασίζονται στο ψάξιμο των εγγραφών με βάση την τιμή που έχει ένα ή περισσότερα από τα πεδία της εγγραφής ή ακόμη και κάποιος συνδυασμός αυτών.

Για όσους ενδιαφέρονται για περισσότερες λεπτομέρειες πάνω στο είδος αυτό οργάνωσης αρχείων, ακολουθεί το πιο κάτω παράδειγμα:

Παράδειγμα:

Ας σκεφτούμε ένα αρχείο που θα χρησιμεύει σαν ατζέντα τηλεφώνων, όπου μεταξύ άλλων, τα πεδία κάθε εγγραφής του είναι:

ΕΠΩΝΥΜΟ
ΟΝΟΜΑ
ΑΡΙΘΜΟΣ 1ου ΤΗΛΕΦΩΝΟΥ

Αν υποθέσουμε ότι απαιτείται από το πρόγραμμα, με το οποίο θα γίνεται η διαχείριση του συγκεκριμένου αρχείου, να προσφέρει τη δυνατότητα αναζήτησης εγγραφών με βάση το "ΕΠΩΝΥΜΟ", τον "ΑΡΙΘΜΟ 1ου ΤΗΛΕΦΩΝΟΥ" ή τον συνδυασμό "ΕΠΩΝΥΜΟ-ΟΝΟΜΑ".
Αυτή είναι μια τυπική περίπτωση αρχείου, όπου απαιτούνται τριών ειδών δείκτες. Άρα πρόκειται για τυπικό παράδειγμα αρχείου με δείκτες.

Η μέθοδος που ακολουθείται, ανάλογα βέβαια με τη χρησιμοποιούμενη γλώσσα προγραμματισμού, είναι η δημιουργία, πέρα από το βασικό αρχείο, και άλλων τριών δευτερευόντων αρχείων δεικτών, κάθε ένα από τα οποία αντιστοιχεί στο πεδίο ή των συνδυασμό των πεδίων, που θα αποτελέσουν τα κλειδιά ή

τους δείκτες αναζήτησης. Κάθε φορά που γίνεται εισαγωγή μιας νέας εγγραφής, παράλληλα με την καταχώρησή της στο βασικό αρχείο, γίνονται και τρεις καταχωρήσεις στα δευτερεύοντα (βοηθητικά) αρχεία, που παίζουν ρόλο δεικτών παραπομπής στο βασικό αρχείο και που ονομάζονται συνήθως αρχεία δεικτών *(index files)*.. Στο πρώτο από αυτά καταχωρείται το "ΕΠΩΝΥΜΟ" μαζί με τη θέση της αντίστοιχης εγγραφής στο βασικό αρχείο. Στο δεύτερο ο "ΑΡΙΘΜΟΣ 1ου ΤΗΛΕΦΩΝΟΥ" και πάλι μαζί με τη θέση της αντίστοιχης εγγραφής στο βασικό αρχείο και, τέλος, στο τρίτο καταχωρίζεται ο συνδυασμός "ΕΠΩΝΥΜΟ - ΟΝΟΜΑ" και, βέβαια, μαζί με τη θέση της αντίστοιχης εγγραφής στο βασικό αρχείο.

Με τον όρο θέση εγγραφής στο αρχείο, ανάλογα με την συγκεκριμένη γλώσσα προγραμματισμού και τη μέθοδο υλοποίησης που επιλέγεται, εννοείται η φυσική διεύθυνση καταχώρησης της εγγραφής στο μέσο αποθήκευσης του αρχείου ή μια λογική έννοια, όπως κάποιος αριθμός (αύξων ή άλλος), που χαρακτηρίζει με μοναδικό τρόπο τη θέση της εγγραφής μέσα στο αρχείο, σε σχέση με τις άλλες εγγραφές.

Όταν μέσα από το πρόγραμμα ζητηθεί το τηλέφωνο του "ΨΩΜΙΑΔΗΣ ΝΙΚΟΛΑΟΣ", τότε αντί να διαβαστεί ολόκληρο το βασικό αρχείο μία προς μία εγγραφή, με μεγάλη πιθανότητα καθυστέρησης, αν ο αριθμός εγγραφών είναι πολύ μεγάλος, αναζητείται με κάποιο κατάλληλο αλγόριθμο γρήγορης αναζήτησης ο επιθυμητός συνδυασμός "ΕΠΩΝΥΜΟ - ΟΝΟΜΑ" στο δευτερεύον αρχείο. Αυτός επισημαίνεται πολύ γρήγορα, γίνεται η παραπομπή στη θέση του βασικού αρχείου όπου φυλάσσεται η πλήρης εγγραφή και στη συνέχεια αυτή ανασύρεται από το αρχείο και τοποθετείται στην κεντρική μνήμη του υπολογιστή για περαιτέρω αξιοποίηση.

Η χρήση δεικτών είναι μια από τις δημοφιλέστερες προγραμματιστικές τεχνικές διαχείρισης αρχείων. Είναι πολύτιμη στην περίπτωση που τα βασικά αρχεία αποτελούνται από πολύ μεγάλο αριθμό εγγραφών, οπότε ο εντοπισμός μιας συγκεκριμένης εγγραφής μέσα σε αυτά είναι πολύ αργός με τη σειριακή αναζήτηση.

Μειονέκτημα της μεθόδου αυτής αποτελεί το γεγονός ότι κάθε φορά που γίνεται οποιαδήποτε μεταβολή (εισαγωγή, τροποποίηση ή διαγραφή εγγραφής), εκτός από το βασικό αρχείο ενημερώνονται όλα τα αρχεία δεικτών, με κόστος στην ταχύτητα απόδοσης του συστήματος. Μάλιστα, σε περιπτώσεις όπου δεν τίθεται ζήτημα ποικιλίας στο είδος των αναζητήσεων και ταυτόχρονα το βασικό αρχείο δεν είναι τόσο μεγάλο, ώστε μια σειριακή αναζήτηση σε αυτό να έχει μεγάλες επιπτώσεις στην ταχύτητα της εφαρμογής, τότε μπορεί ο σειριακός τρόπος οργάνωσης να αποδειχτεί αποτελεσματικός και, ίσως, αποτελεσματικότερος από την οργάνωση με δείκτες.

Σε πάρα πολλές περιπτώσεις μπορεί να χρησιμοποιηθεί και ένα σύστημα οργάνωσης αρχείων πιο αποτελεσματικό ως προς την ταχύτητα αναζήτησης και ανεύρεσης στοιχείων σε σύγκριση με αυτά που παρουσιάστηκαν παραπάνω. Πρόκειται για μια μέθοδο οργάνωσης και πρόσβασης σε αρχεία, που είναι γνωστή σαν **μέθοδος τυχαίας προσπέλασης** *(random access)*.

Για την κατανόηση του τρόπου αποθήκευσης και αναζήτησης δεδομένων με τη μέθοδο της τυχαίας προσπέλασης, παραθέτουμε το παρακάτω παράδειγμα:

Παράδειγμα:

Ας υποθέσουμε ότι σε ένα μικρό βιβλιοπωλείο η διαχείριση της αποθήκης των βιβλίων είναι ένα αυτόνομο σύστημα, που χρησιμοποιεί μόνο ένα αρχείο για να κρατά πληροφορίες για τα βιβλία που διακινούνται. Η ενημέρωση του αρχείου και η αναζήτηση συγκεκριμένων βιβλίων βασίζεται σε ένα μοναδικό για κάθε βιβλίο κωδικό. Ο κωδικός επιλέγεται με τέτοιο τρόπο, ώστε σε κάθε βιβλίο να αντιστοιχεί ένας αριθμός μεγαλύτερος από το 0, έτσι ώστε αν π.χ., το τελευταίο καταχωρημένο βιβλίο έχει σαν κωδικό τον αριθμό 1000, το επόμενο να πάρει το 1001. Στη συγκεκριμένη περίπτωση μπορεί να χρησιμοποιηθεί η εξής τεχνική οργάνωσης των εγγραφών του αρχείου:

Αν το μήκος εγγραφής είναι 100 bytes και στο πρώτο καταχωρημένο βιβλίο αποδοθεί ο κωδικός 0, τότε η πρώτη εγγραφή ξεκινά από τη σχετική διεύθυνση 0 του αρχείου και καταλαμβάνει τις θέσεις από 0 ως 99, η δεύτερη ξεκινά από τη φυσική διεύθυνση 100 μέχρι την 199, η επόμενη από την 200 ως 299 και γενικά η ν εγγραφή ξεκινά από τη σχετική διεύθυνση (ν-1)*100. Αν λοιπόν θέλουμε να ενημερώσουμε το αρχείο για κάποια αλλαγή που αφορά το βιβλίο με κωδικό 130, πληκτρολογούμε στη σχετική οθόνη στη θέση του κωδικού τον αριθμό αυτό, το πρόγραμμα πηγαίνει αμέσως στη διεύθυνση (130-1)*100=12900 του αρχείου και ανασύρει στη μνήμη για επεξεργασία τη συγκεκριμένη εγγραφή.

Το παράδειγμα αυτό παρουσιάζει την απλούστερη εκδοχή ενός τρόπου οργάνωσης και πρόσβασης σε αρχεία δεδομένων, η οποία είναι γνωστή σαν μέθοδος τυχαίας προσπέλασης *(random access)* στα δεδομένα. Με βάση αυτή την εκδοχή, υπάρχει άμεση αντιστοίχιση της λογικής με τη φυσική αποθήκευση των δεδομένων στο αρχείο. Στην πράξη βέβαια, χρησιμοποιούνται και άλλες περισσότερο πολύπλοκες παραλλαγές της μεθόδου αυτής, ανάλογα και με τα εργαλεία προγραμματισμού που επιλέγονται κατά περίπτωση, αλλά η παράθεσή τους ξεφεύγει από τα όρια του βιβλίου αυτού. Το βέβαιο είναι πάντως, πως από τις διάφορες μεθόδους προσπέλασης σε αρχεία δεδομένων, ταχύτερες είναι εκείνες που χρησιμοποιούν κάποια παραλλαγή των πιο πάνω τεχνικών. Ο λόγος είναι ότι με χρήση μιας τέτοιας τεχνικής, το πρόγραμμα γνωρίζει πού ακριβώς θα βρει την εγγραφή μέσα στο αρχείο και το σύστημα ανάγνωσης του μέσου αποθήκευσης τοποθετείται άμεσα στο συγκεκριμένο σημείο και ανασύρει την εγγραφή που ζητείται.

"Πες μου τα προβλήματα, που θέλεις να αντιμετωπίσεις και το υπόβαθρό σου στον προγραμματισμό εφαρμογών, να σου πω, ποιο εργαλείο ανάπτυξης να επιλέξεις" Αυτό θα μπορούσε να είναι το αξίωμα επιλογής προγραμματιστικού εργαλείου για την ανάπτυξη μιας εφαρμογής.

Η δυνατότητα τυχαίας προσπέλασης σε ένα αρχείο δεν υποστηρίζεται από όλες τις γλώσσες προγραμματισμού. Καταλληλότερες είναι γλώσσες με ευελιξία στη διαχείριση δεδομένων όπως C, C++, Pascal κ.α. Οι γλώσσες αυτές είναι κατάλληλες και για τη σειριακή διαχείριση αρχείων, αλλά από κατασκευής δεν είναι ιδιαίτερα επιδέξιες στη διαχείριση αρχείων με δείκτες. Για την τελευταία αυτή κατηγορία είναι ιδανική η Cobol, η οποία, όμως, πέρα από την αποτελεσματική διαχείριση αρχείων, δεν έχει μεγάλη ευελιξία στην αντιμετώπιση άλλων σύγχρονων προγραμματιστικών προβλημάτων, εξαιτίας της ξεπερασμένης φιλοσοφίας της. Βέβαια, θα πρέπει να σημειωθεί ότι τα τελευταία χρόνια γίνεται μια πολύ σοβαρή προσπάθεια εκσυγχρονισμού της γλώσσας Cobol, γεγονός που οφείλεται στη μεγάλη εγκατεστημένη βάση εφαρμογών λογισμικού που είναι γραμμένες στη γλώσσα αυτή. Γενικά, δεν υπάρχει το απόλυτο προγραμματιστικό εργαλείο για τη διαχείριση αρχείων, αλλά εργαλεία περισσότερο ή λιγότερα κατάλληλα για συγκεκριμένου τύπου προβλήματα.

Οι τρόποι οργάνωσης που περιγράφηκαν παραπάνω είναι οι επικρατέστεροι, αλλά όχι οι μόνοι. Στην πράξη υπάρχουν αρκετοί αλγόριθμοι, που υλοποιούν περισσότερο σύνθετες μεθόδους διαχείρισης αρχείων, κατάλληλες για την αντιμετώπιση πραγματικών προβλημάτων. Όσο για τα εργαλεία που χρησιμοποιεί κάθε γλώσσα, οι όποιοι περιορισμοί υπάρχουν σήμερα, αντιμετωπίζονται συνήθως με βιβλιοθήκες έτοιμου κώδικα, οι οποίες προσφέρονται από εταιρίες κατασκευής εργαλείων προγραμματισμού και λειτουργούν ως συμπλήρωμα των εργαλείων που διαθέτει από την κατασκευή της η κάθε γλώσσα. Με αυτόν τον τρόπο αντιμετωπίζονται οι αδυναμίες κάθε μίας από αυτές σε συγκεκριμένους τομείς.

Βασικό μειονέκτημα στην παραδοσιακή οργάνωση των δεδομένων σε αρχεία, είναι ότι η προσπέλαση των δεδομένων γίνεται μόνο με τα συγκεκριμένα προγράμματα, που έχουν δημιουργηθεί για την εφαρμογή. Το μειονέκτημα αυτό δεν υπάρχει στα ΣΔΒΔ, όπου υπάρχει ανεξαρτησία των δεδομένων από τα προγράμματα μιας εφαρμογής.

6.1.3. Σύγχρονοι τρόποι παρουσίασης των δεδομένων

Τα αρχεία αποθηκεύουν δεδομένα. Το πώς παρουσιάζονται τα δεδομένα αυτά στους χρήστες τους εξαρτάται από τις απαιτήσεις του συστήματος και τις δυνατότητες του προγράμματος. Η παρουσίαση των δεδομένων των αρχείων γίνεται μέσα από εργαλεία, που σε κάποιες περιπτώσεις αποτελούν μέρος της γλώσσας προγραμματισμού που χρησιμοποιείται, ενώ σε άλλες λειτουργούν συμπληρωματικά.

Τα σύγχρονα περιβάλλοντα ανάπτυξης επιτρέπουν τη συνεργασία και ανταλλαγή δεδομένων μεταξύ διαφορετικών προγραμμάτων. Αξιοποιώντας αυτή τη δυνατότητα μπορεί κανείς να χρησιμοποιεί για παράδειγμα μια γλώσσα προγραμματισμού για τη δημιουργία προγραμμάτων εισαγωγής δεδομένων σε ένα αρχείο, παράλληλα με τη χρήση άλλου προγράμματος δημιουργίας εκτυπώσιμων αναφορών *(reports)*. Είναι μάλιστα δυνατό η συνεργασία των διαφορετικών προγραμμάτων να γίνεται με τέτοιο τρόπο, ώστε ο τελικός χρήστης να μην αντιλαμβάνεται την ύπαρξη των διαφορετικών προγραμμάτων. Μπορεί π.χ., το πρόγραμμα που παράγει τις εκτυπώσεις να καλείται μέσα από το άλλο πρόγραμμα, ενεργοποιούμενο με το πάτημα ενός "κουμπιού" με το ποντίκι.

Τα εξειδικευμένα προγράμματα δημιουργίας εκτυπώσεων είναι συνήθως γνωστά σαν γεννήτριες αναφορών *(report generators)* και είναι σήμερα ιδιαίτερα δημοφιλή, γιατί επιτρέπουν τον ανεξάρτητο σχεδιασμό των εκτυπώσιμων αναφορών και στη συνέχεια τη συνεργασία τους με το βασικό πρόγραμμα της εφαρμογής. Ο σχεδιασμός των εκτυπώσεων γίνεται με εύκολο, γρήγορο και ευχάριστο τρόπο με χρήση του ποντικιού, σε αντίθεση με τον παραδοσιακό τρόπο δημιουργίας εκτυπώσεων με χρήση εντολών, που δεν παρείχαν στον προγραμματιστή τη δυνατότητα να βλέπει άμεσα το αποτέλεσμα αυτού που δημιουργεί, παρά μόνο κατά τη φάση του ελέγχου και των δοκιμών. Όλα τα σύγχρονα περιβάλλοντα δημιουργίας εφαρμογών (C++, Pascal, Java, Visual Basic κ.λ.π), είτε διαθέτουν ενσωματωμένες γεννήτριες αναφορών, είτε συνεργάζονται με άλλες. Στα περιβάλλοντα ανάπτυξης μεγάλων εφαρμογών όπως τα συστήματα διαχείρισης βάσεων δεδομένων (π.χ., Oracle, Sybase, Informix, DB2, Ingres), η ύπαρξη γεννητριών εφαρμογών θεωρείται αυτονόητη.
Πέρα από τη σύνδεση με γεννήτριες εφαρμογών, τα προγράμματα που διαχειρίζονται αρχεία μπορούν να μεταβιβάσουν δεδομένα και σε προγράμματα επεξεργασίας κειμένου ή προγράμματα δημιουργίας λογιστικών φύλλων, που προσφέρουν εναλλακτικούς τρόπους παρουσίασης των δεδομένων τους.

Πολύ διαδεδομένη στις σύγχρονες εφαρμογές είναι και η δυνατότητα παρουσίασης στοιχείων στο διαδίκτυο σε μορφή HTML ή PDF (βλ. κεφ. 1), που αποτελούν ευρέως αποδεκτά πρότυπα που χρησιμοποιούνται στον παγκόσμιο ιστό (World Wide Web). Τα δύο αυτά πρότυπα υποστηρίζονται από τα περισσότερα προγράμματα δημιουργίας αναφορών, αντικαθιστώντας σταδιακά τις πιο παραδοσιακές μορφές εκτυπώσεων, αφού είναι αναγνώσιμες από τον τελικό χρήστη, με τη χρήση του προγράμματος περιήγησης που χρησιμοποιεί για την πρόσβαση στο διαδίκτυο *(browser)*, ή με τη χρήση ενός δημοφιλούς προγράμματος ανάγνωσης PDF αρχείων (π.χ., Acrobat Reader).

6.2. ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΚΑΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

6.2.1. Φιλοσοφία και βασικές αρχές των βάσεων δεδομένων

Η ομαδοποίηση των δεδομένων σε πεδία, εγγραφές και αρχεία, συνιστά τον παραδοσιακό τρόπο οργάνωσης και αποθήκευσής τους και παραμένει η επικρατούσα λύση στα σύγχρονα αυτοματοποιημένα συστήματα. Όπως έχει προαναφερθεί στο κεφάλαιο 1, βασικό μειονέκτημα αυτού του τρόπου αποθήκευσης των δεδομένων είναι ότι εξαρτάται αποκλειστικά και μόνο από τις συγκεκριμένες ανάγκες των τωρινών προγραμμάτων του συστήματος, αφού αν ο χρήστης διαπιστώσει κάποια στιγμή, ότι χρειάζεται και ένα διαφορετικό τρόπο παρουσίασης των στοιχείων, ή κάποιο συνδυασμό δεδομένων από διαφορετικά αρχεία, πρέπει να γραφτεί νέος και κάποτε πολύπλοκος κώδικας. Η διαδικασία αυτή στοιχίζει σε χρόνο και χρήμα και βλάπτει τη λειτουργικότητα του συστήματος. Στην ουσία, ακόμη και η πιο ασήμαντη αλλαγή στην παρουσίαση πληροφοριών απαιτεί το γράψιμο νέου κώδικα.

Από την άλλη πλευρά, αυτή η εξάρτηση των δεδομένων από το πρόγραμμα, παρέχει δυνατότητες στους ικανούς προγραμματιστές να χρησιμοποιήσουν αλγόριθμους οργάνωσης των αρχείων, προσανατολισμένους στις συγκεκριμένες ανάγκες, πετυχαίνοντας πολύ καλούς χρόνους προσπέλασης των αρχείων και συνολικά καλύτερη απόκριση του συστήματος.

Τα τελευταία χρόνια κερδίζει έδαφος μια άλλου είδους προσέγγιση στην οργάνωση των δεδομένων, προσανατολισμένη στα ίδια τα δεδομένα, με δυνατότητες δυναμικής δημιουργίας συνδυασμών και παρουσίασης τους, ακόμα και από μη προγραμματιστές. Η προσέγγιση αυτή υλοποιείται από ειδικά συστήματα λογισμικού γνωστά σαν Συστήματα Διαχείρισης Βάσεων Δεδομένων *(Database Management Systems)* ή για λόγους συντομίας ΣΔΒΔ *(DBMS)* (βλ. και κεφ. 1,2,3).

Τα συστήματα αυτά προσφέρουν μια πληθώρα εργαλείων για την ανάπτυξη εφαρμογών, όπως εργαλεία για σχεδιασμό οθονών εισαγωγής στοιχείων, εκτυπώσιμων αναφορών, διαχείρισης δεδομένων χωρίς ειδικές γνώσεις προγραμματισμού, παρουσίασης δεδομένων στο διαδίκτυο, σύνδεσής τους με δεδομένα άλλων συστημάτων κ.λ.π.

Ορισμένα από τα ΣΔΒΔ είναι προσανατολισμένα στην ανάπτυξη εφαρμογών για προσωπικούς υπολογιστές ή δίκτυα προσωπικών υπολογιστών (Microsoft Access, Dbase, Paradox κ.α.), ενώ

κάποια άλλα αφορούν περισσότερο στην ανάπτυξη μεγάλων εφαρμογών, κατάλληλων για σύνθετα περιβάλλοντα, που αποτελούνται από πολλούς και με διάφορους τρόπους διασυνδεδεμένους, υπολογιστές (Oracle, Sybase, Ingress, Informix, SQL Server, DB2 κ.α.).

Η πρόσβαση σε κυβερνητικές, στρατιωτικές και εμπορικές βάσεις δεδομένων υπόκειται σε περιορισμούς, που προστατεύουν τα πολύτιμα δεδομένα από ανεπιθύμητη πρόσβαση, ειδικά σήμερα, που οι μεγάλοι οργανισμοί επιτρέπουν τη χρήση κάποιων από τα αποθηκευμένα σε βάσεις δεδομένα τους, μέσα από το διαδίκτυο.

Τα σύγχρονα ΣΔΒΔ ενσωματώνουν πολλαπλές δυνατότητες διασφάλισης των δεδομένων από ανεπιθύμητη πρόσβαση, δυνατότητες που εναπόκειται σε αυτούς που αναπτύσσουν, ή διαχειρίζονται το σύστημα να τις ενεργοποιήσουν. Τα ΣΔΒΔ επίσης, επιτρέπουν την κατανομή των δεδομένων ακόμη και σε διαφορετικά υπολογιστικά συστήματα, που μπορεί να βρίσκονται σε διαφορετικούς φυσικούς χώρους, ώστε να αξιοποιούν τις δυνατότητες των σύγχρονων δικτύων, βελτιστοποιώντας την απόδοσή τους. Στην περίπτωση αυτή, οι βάσεις ονομάζονται κατανεμημένες *(distributed)*.

Η τεράστια τεχνολογική εξέλιξη των ΣΔΒΔ την τελευταία δεκαετία οδήγησε στη μεγάλη διάδοσή τους. Σήμερα, οποιαδήποτε εφαρμογή, έστω και μικρομεσαίου μεγέθους, προσανατολισμένη κυρίως στη διαχείριση δεδομένων, έχει ως πρώτη επιλογή από τα εργαλεία ανάπτυξης, κάποιο ΣΔΒΔ, έστω και περιορισμένων δυνατοτήτων.

6.2.2. Βασικές έννοιες και αρχές των σχεσιακών ΣΔΒΔ

Σε αντίθεση με τα προσανατολισμένα στη διαχείριση αρχείων συστήματα, οι βάσεις δεδομένων χρησιμοποιούν τον εξής τρόπο οργάνωσης:

Αρχικά ανιχνεύονται από το περιβάλλον στο οποίο θα εγκατασταθεί το σύστημα τα εξής:

- Οι **οντότητες** *(entities)* που θα το συνθέσουν, δηλαδή πρόσωπα, πράξεις, γεγονότα και έννοιες γύρω από τις οποίες θα αποθηκευτούν πληροφορίες. Για λόγους καλύτερης κατανόησης, θα μπορούσε να τις χαρακτηρίσει κανείς σαν το λογικό αντίστοιχο της εγγραφής ενός αρχείου.
- Οι **ιδιότητες** *(attributes)*, οι οποίες χαρακτηρίζουν τις οντότητες (λογικό αντίστοιχο, τα πεδία της εγγραφής ενός αρχείου).
- Οι **σχέσεις** *(relations)* μεταξύ των οντοτήτων. Ο συσχετισμός των οντοτήτων με βάση κάποιες καθορισμένες σχέσεις, είναι το χαρακτηριστικό των σχεσιακών βάσεων δεδομένων.

Παράδειγμα:

1. Σε ένα σύστημα που διαχειρίζεται τα στοιχεία των μαθητών ενός σχολείου, **ο μαθητής** είναι μια οντότητα με ιδιότητες όπως κωδικός, ονοματεπώνυμο, όνομα πατρός, τμήμα στο οποίο υπάγεται, βαθμολογία κατά μάθημα κ.λ.π.

2. **Το τιμολόγιο** μιας εταιρίας λιανικής πώλησης είναι οντότητα με ιδιότητες όπως ο αριθμός τιμολογίου, το ποσό, η ημερομηνία έκδοσης κ.λ.π.

3. **Ο υπάλληλος μιας εταιρίας** είναι οντότητα με ιδιότητες όπως ο κωδικός υπαλλήλου, το ονοματεπώνυμο, η ειδικότητα του υπαλλήλου, η θέση του στην εταιρία, ο μισθός του, ο κωδικός του τμήματος στο οποίο είναι ενταγμένος κ.α.

Πίσω από τα σχεσιακά συστήματα βάσεων δεδομένων βρίσκεται μια ολόκληρη θεωρία με σοβαρές μαθηματικές αναφορές. Πατέρας αυτής της θεωρίας που ονομάζεται "σχεσιακή" και πρωτοεμφανίστηκε στις αρχές της δεκαετίας του 1970, είναι ο **E.F. Codd,** του οποίου μάλιστα η εκτίμηση ήταν ότι το **σχεσιακό πρότυπο** θα επικρατήσει τελικά πλήρως, πράγμα που επιβεβαιώθηκε σε σημαντικό βαθμό από την πραγματικότητα της σημερινής αγοράς των βάσεων δεδομένων. Σύμφωνα με τον Codd, οι οντότητες και οι σχέσεις οντοτήτων παριστάνονται με τον ίδιο τρόπο σαν **σχέσεις πινάκων.**

Αν και οι παραπάνω έννοιες πρωτοαναφέρθηκαν στο κεφάλαιο 1, για να γίνουν καλύτερα κατανοητές και προκειμένου να υπάρξει εξοικείωση με τον σχετικό συμβολισμό που χρησιμοποιείται για την απεικόνισή τους, παρουσιάζονται τα πιο κάτω παραδείγματα:

Σε ένα σύστημα παρακολούθησης παραγγελιών ενός καταστήματος λιανικής πώλησης, δύο από τις οντότητες που μπορούμε να διακρίνουμε είναι οι εξής:

ΠΡΟΜΗΘΕΥΤΗΣ (ΚΩΔΙΚΟΣ, ΠΟΛΗ_ΠΟΥ_ΕΔΡΕΥΕΙ, ΤΗΛΕΦΩΝΟ, ΤΑΧ_ΚΩΔΙΚΑΣ, ΑΡΙΘ_FAX).

Με αυτό τον τρόπο συμβολίζουμε μια οντότητα (ΠΡΟΜΗΘΕΥΤΗΣ) και σε παρενθέσεις τις ιδιότητες της οντότητας. Πολλές φορές η ίδια η οντότητα ονομάζεται και σχέση (σύμφωνα με τη σχεσιακή θεωρία).

Παράδειγμα:

ΕΙΔΟΣ (ΚΩΔΙΚΟΣ, ΟΝΟΜΑΣΙΑ, ΔΙΑΘΕΣΙΜΗ_ΠΟΣΟΤΗΤΑ, ΚΩΔΙΚΟΣ_ΠΡΟΜΗΘΕΥΤΗ).

Η οντότητα (ή σχέση) "ΕΙΔΟΣ", έχει τις ιδιότητες που δηλώνονται με συμβολικό τρόπο μέσα στην παρένθεση. Παρατηρήστε ότι μία από τις ιδιότητες ("ΚΩΔΙΚΟΣ_ΠΡΟΜΗΘΕΥΤΗ") αναφέρεται σε ένα κωδικό υπαρκτού προμηθευτή, έτσι ώστε κάθε είδος που υπάρχει στο κατάστημα να **συσχετίζεται** με τον προμηθευτή, ο οποίος διαθέτει το συγκεκριμένο είδος.

Στην ουσία λοιπόν η **σχέση** μεταξύ των δύο οντοτήτων είναι ο σύνδεσμος που επιτυγχάνεται με την ιδιότητα "ΚΩΔΙΚΟΣ_ΠΡΟΜΗΘΕΥΤΗ", που παραπέμπει στην οντότητα "ΠΡΟΜΗΘΕΥΤΗΣ". Οι πίνακες του πιο κάτω **σχήματος 6.2** παρουσιάζουν με απλό τρόπο τη σχέση μεταξύ συγκεκριμένων ειδών και προμηθευτών με κλειδί συσχετισμού την ιδιότητα "ΚΩΔΙΚΟΣ_ΠΡΟΜΗΘΕΥΤΗ". Σε κάθε προμηθευτή του παραδείγματος αντιστοιχούν ένα ή περισσότερα είδη. Υπάρχει ιδιότητα που ονομάζεται "ΚΩΔΙΚΟΣ", τόσο στη οντότητα "ΕΙΔΟΣ", όσο και στην οντότητα "ΠΡΟΜΗΘΕΥΤΗΣ" και συμπτωματικά και στις δύο περιπτώσεις η ιδιότητα αυτή χαρακτηρίζει με μοναδικό τρόπο την κάθε μια, αποτελώντας ουσιαστικά την ταυτότητα των οντοτήτων για κάθε ξεχωριστή γραμμή δεδομένων. Κάθε μια από αυτές τις γραμμές ονομάζεται **πλειάδα** *(tuple)* και είναι το λο-

γικό αντίστοιχο της εγγραφής, όπως την ξέρουμε από τα αρχεία. Η ιδιότητα "ΚΩΔΙΚΟΣ", που χαρακτηρίζει με μοναδικό τρόπο κάθε πλειάδα και στις δύο οντότητες, αποτελεί το **πρωτεύον κλειδί** *(primary key)*, της οντότητας.

Το **Πρωτεύον κλειδί** απαγορεύεται να έχει πολλαπλές τιμές. Με τον τρόπο αυτό χαρακτηρίζει μονοσήμαντα κάθε εγγραφή μέσα σε ένα αρχείο.

ΕΙΔΟΣ

ΚΩΔΙΚΟΣ	ΟΝΟΜΑΣΙΑ	ΔΙΑΘΕΣΙΜΗ ΠΟΣΟΤΗΤΑ	ΚΩΔΙΚΟΣ ΠΡΟΜΗΘΕΥΤΗ
1111	ΜΟΛΥΒΙ	1002	02
2222	ΓΟΜΑ	200	06
3333	ΞΥΣΤΡΑ	3234	02
4444	ΣΤΥΛΟ	2378	01
5555	ΤΕΤΡΑΔΙΟ	567	01

ΠΡΟΜΗΘΕΥΤΗΣ

ΚΩΔΙΚΟΣ	ΠΟΛΗ_ ΠΟΥ_ ΕΔΡΕΥΕΙ	ΤΗΛΕ ΦΩΝΟ	ΤΑΧ ΚΩΔΙΚΑΣ	ΑΡΙΘ FAX
01	Αθήνα	3233677	11521	3233678
02	Αθήνα	2333766	11245	2366777
03	Λάρισα	888867	10453	887879
04	Πάτρα	556768	12345	557879
05	Πάτρα	557899	10934	557888
06	Αθήνα	6898777	11257	6899777

Σχήμα 6.2 – Συσχετισμός οντοτήτων.

Ο ορισμός πρωτεύοντος κλειδιού, το οποίο άλλωστε αποτελεί την ταυτότητα μιας οντότητας είναι σε κάθε περίπτωση υποχρεωτικός. Το πρωτεύον κλειδί επιβάλλεται να έχει πάντα τιμή, ενώ απαγορεύεται να έχει πολλαπλές τιμές, π.χ., δεν μπορεί να υπάρχουν δύο πλειάδες της οντότητας "ΠΡΟΜΗΘΕΥΤΗΣ" με την ίδια τιμή "01" στο πρωτεύον κλειδί.

Η ιδιότητα "ΚΩΔΙΚΟΣ_ΠΡΟΜΗΘΕΥΤΗ", η οποία υλοποιεί τη συσχέτιση των δύο οντοτήτων, αποτελεί **ξένο κλειδί** *(foreign key)* για την οντότητα "ΕΙΔΟΣ", δηλαδή παραπέμπει στο πρωτεύον κλειδί μιας άλλης οντότητας, στη συγκεκριμένη περίπτωση της οντότητας "ΠΡΟΜΗΘΕΥΤΗΣ". Γενικά, το ξένο κλειδί μιας οντότητας μπορεί να έχει πολλαπλές τιμές, όπως άλλωστε συμβαίνει στο πιο πάνω παράδειγμα. Επίσης, αν η οντότητα συσχετίζεται με παραπάνω από μία άλλες οντότητες, προφανώς περιέχει ξένο κλειδί για κάθε μία από αυτές. Το πρωτεύον (όπως και το ξένο) κλειδί μπορεί να είναι σύνθετο *(compound)*, δηλαδή να αποτελεί σύνθεση περισσοτέρων της μιας ιδιοτήτων.

Για παράδειγμα, σε μια οντότητα (π.χ., "ΜΑΘΗΤΗΣ"), που έχει ως ιδιότητες το επώνυμο, το όνομα και το όνομα πατρός του μαθητή, θα μπορούσε να επιλέξει κανείς ως πρωτεύον κλειδί το συνδυασμό επώνυμο, όνομα, όνομα πατρός, που πιθανότατα χαρακτηρίζει με μοναδικό τρόπο τον κάθε μαθητή. Η σήμανση κάθε ιδιότητας που αποτελεί ή είναι μέρος πρωτεύοντος κλειδιού, γίνεται συνήθως με το σύμβολο "#" Π.χ., ο συμβολισμός των οντοτήτων "ΕΙΔΟΣ" και "ΠΡΟΜΗΘΕΥΤΗΣ" θα μπορούσε να διαμορφωθεί με την επισήμανση του πρωτεύοντος κλειδιού όπως πιο κάτω:
ΠΡΟΜΗΘΕΥΤΗΣ(ΚΩΔΙΚΟΣ#,ΠΟΛΗ_ΠΟΥ_ΕΔΡΕΥΕΙ,ΤΗΛΕΦΩΝΟ, ΤΑΧ_ΚΩΔΙΚΑΣ, ΑΡΙΘ_FAX).

ΕΙΔΟΣ(ΚΩΔΙΚΟΣ#,ΟΝΟΜΑΣΙΑ,ΔΙΑΘΕΣΙΜΗ_ΠΟΣΟΤΗΤΑ,ΚΩΔΙΚΟΣ_ΠΡΟΜΗΘΕΥΤΗ)

Η περιγραφή του συνόλου επιτρεπτών τιμών (πεδίο ορισμού) μιας ιδιότητας γίνεται με τη βοήθεια μαθηματικών συμβολισμών, που παραπέμπουν στη θεωρία συνόλων. Έτσι, π.χ., ο συμβολισμός που περιγράφει από πού παίρνει τιμές η ιδιότητα "ΚΩΔΙΚΟΣ" της οντότητας "ΠΡΟΜΗΘΕΥΤΗΣ" είναι ο εξής:

DOM(ΚΩΔΙΚΟΣ) = {01,02,03,04?}.

Μπορούν να χρησιμοποιηθούν όλα τα σύμβολα της θεωρίας συνόλων, αν αυτά μπορούν να βοηθήσουν στην περιγραφή του πεδίου ορισμού μιας ιδιότητας. Στην περίπτωση του συσχετισμού των οντοτήτων "ΕΙΔΟΣ" και "ΠΡΟΜΗΘΕΥΤΗΣ", όπως περιγράφηκε παραπάνω, σε κάθε προμηθευτή μπορεί να αντιστοιχούν πολλά είδη προϊόντων. Στην περίπτωση αυτή η σχέση "ΕΙΔΟΣ-ΠΡΟΜΗΘΕΥΤΗΣ" ονομάζεται **σχέση πολλά προς ένα** *(many-to-one relationship).* Σε άλλη περίπτωση θα μπορούσε σε κάθε τιμή του πρωτεύοντος κλειδιού της οντότητας "ΟΝΤΟΤΗΤΑ1" να αντιστοιχεί μόνο μία τιμή για το ξένο κλειδί μιας άλλης οντότητας, της "ΟΝΤΟΤΗΤΑ2". Η σχέση της "ΟΝΤΟΤΗΤΑ1" με την "ΟΝΤΟΤΗΤΑ2" (προσοχή στη σειρά με την οποία αναφέρονται οι οντότητες), είναι **σχέση ένα προς ένα** *(one-to-one relationship).*

Με βάση αυτή τη λογική, οι σχέσεις μεταξύ οντοτήτων μπορούν να χωριστούν στις εξής κατηγορίες:

- **1 προς 1** *(one-to-one)*
- **1 προς Ν** *(one-to-many)*
- **Ν προς 1** *(many-to-one)*
- **Μ προς Ν** *(many-to-many)*

όπου Ν, Μ συμβολίζουν την ύπαρξη της έννοιας "πολλά" στο συσχετισμό των οντοτήτων.

Ο **πίνακας 6.1** που ακολουθεί, βοηθά στην κατανόηση των διαφορετικών αυτών συσχετίσεων:

ΠΟΛΗ ΠΡΩΤΕΥΟΥΣΑ - ΝΟΜΟΣ Η ΠΟΛΗ Χ είναι μοναδική πρωτεύουσα του ΝΟΜΟΥ Ψ
Σχέση (1:1) ΑΝΘΡΩΠΟΣ - ΣΠΙΤΙ ΑΝΘΡΩΠΟΣ(ΟΙ) ιδιοκτήτης(ες) ΣΠΙΤΙΟΥ(ΩΝ)
Σχέση (Μ:Ν) ΣΧΟΛΙΚΟ ΤΜΗΜΑ - ΜΑΘΗΤΕΣ Κάθε ΣΧΟΛΙΚΟ ΤΜΗΜΑ έχει πολλούς ΜΑΘΗΤΕΣ Σχέση (1:Μ)
ΥΠΑΛΛΗΛΟΣ - ΤΜΗΜΑ ΥΠΑΛΛΗΛΟΣ(ΟΙ) ανήκει(ουν) σε ένα μόνο ΤΜΗΜΑ Σχέση (Μ:1)

Πίνακας 6.1. – Κατηγορίες συσχετίσεων.

Σημείωση:

Στο 2ο μάθημα του τετραδίου εργασίας, συνεχίζεται η δουλειά με το παραδειγματικό σύστημα έκδοσης αεροπορικών εισιτηρίων, με το σταδιακό πέρασμα από τη φάση της ανάλυσης στη φάση της υλοποίησης του συστήματος. Ζητείται συγκεκριμένα η δημιουργία των οντοτήτων στις οποίες θα βασιστεί το σύστημα, εργασία η οποία είναι πολύ ενδιαφέρουσα και εισάγει στις έννοιες του σχεδιασμού μιας βάσης δεδομένων.

6.2.3. Κανονικοποίηση και διαγράμματα οντοτήτων - συσχετίσεων

Στη διαδικασία του σχεδιασμού ενός **σχεσιακού συστήματος βάσεων δεδομένων** (ΣΣΔΒΔ - RDBMS) η σημαντικότερη ίσως, φάση, είναι αυτή της κανονικοποίησης των οντοτήτων.

Ορισμός

Κανονικοποίηση *(normalization)* είναι η τυπική διαδικασία, η οποία ομαδοποιεί τα δεδομένα σε μία φόρμα κατάλληλη, για να αυξήσει την απόδοση του ΣΔΒΔ, να μπορεί να αντέξει στο χρόνο και να ελαχιστοποιήσει την επίδραση όποιων μελλοντικών αλλαγών στη σταθερότητα του συστήματος.

Στην ουσία, με τη διαδικασία της κανονικοποίησης των δεδομένων επιτυγχάνεται η καλύτερη δυνατή οργάνωσή τους, άρα και ο καλύτερος δυνατός σχεδιασμός της βάσης δεδομένων που θα προκύψει.

Καλύτερος δυνατός σχεδιασμός από την άποψη των χρηστών της βάσης σημαίνει:

- Σχέσεις μεταξύ των δεδομένων (οντοτήτων) εύκολα κατανοητές και τροποποιήσιμες.
- Κάθε καινούρια απαίτηση από το σύστημα που θα διαχειριστεί τα δεδομένα να μπορεί να ενσωματωθεί εύκολα.
- Να υπάρχουν έλεγχοι και μηχανισμοί αποτροπής της αποθήκευσης ακατάλληλων δεδομένων.
- Να υπάρχει πρόβλεψη για την κατάλληλη ομαδοποίηση και την ταχύτερη δυνατή προσπέλαση των δεδομένων από το λογισμικό διαχείρισης των δεδομένων και τον απαιτούμενο εξοπλισμό (υπολογιστές).

Πέρα από αυτές τις ανάγκες η διαδικασία της κανονικοποίησης έχει ως στόχο να αντιμετωπίσει τα διάφορα προβλήματα, τα οποία σχετίζονται με τον τρόπο, με τον οποίο θα αποθηκευτούν τα δεδομένα στη βάση και που μπορεί, αν δεν προβλεφτούν από την αρχή σωστά, να οδηγήσουν σε δυσλειτουργίες, που είναι γνωστές ως **προβλήματα (ή ανωμαλίες) τροποποιήσεων** *(modification anomalies)*. Με τη σωστή κανονικοποίηση των δεδομένων επιτυγχάνονται και τα εξής:

- Αποτρέπεται η περιττή επανάληψη δεδομένων.
- Εξασφαλίζεται η συνοχή και η σωστή συνδεσιμότητα των δεδομένων μεταξύ τους.
- Εξασφαλίζεται η απλούστερη δυνατή οργάνωση των δεδομένων και η δυνατότητα για μελλοντικές τροποποιήσεις, χωρίς μεγάλες επιπτώσεις στη συνολική λειτουργία του συστήματος.

Για την καλύτερη δυνατή κατανόηση των αναγκαιοτήτων και των εννοιών, που σχετίζονται με την κανονικοποίηση των δεδομένων, θα επιχειρηθεί η παρουσίαση των βασικότερων από αυτές, μέσα από το παράδειγμα που ακολουθεί:

Ο ιδιοκτήτης ενός φροντιστηρίου αποφασίζει να βάλει σε μία βάση δεδομένων τους μαθητές του, ώστε σε κάθε περίπτωση να μπορεί να γνωρίζει τα βασικά στοιχεία και τις επιδόσεις

τους. Ξεκινά από ένα φύλλο χαρτί, όπου προσπαθώντας να βάλει σε μια σειρά τα στοιχεία που μέχρι τώρα έχει καταχωρημένα σε απλές καρτέλες, κατασκευάζει ένα πίνακα με τα στοιχεία ορισμένων από αυτούς, ώστε να δει πώς θα μπορούσε να τα οργανώσει καλύτερα.

Παράδειγμα:

Από τον πιο κάτω πίνακα 6.2 γίνονται φανερά τα εξής: Υπάρχει επανάληψη πληροφοριών στα "Υπόλοιπα στοιχεία" και στα "Αποτελέσματα στα τεστ των μαθημάτων". Συγκεκριμένα επαναλαμβάνεται συνέχεια το φύλο του κάθε μαθητή καθώς και τα μαθήματα, τα οποία επιλέγουν οι μαθητές.

Κωδικός μαθητή	***Επώνυμο***	***Υπόλοιπα στοιχεία***	***Αποτελέσματα στα τεστ των μαθημάτων***
010	Πέτρου	Αγόρι, Ημ.γέννησης: 07/05/82	ΜΑΘΗΜΑΤΙΚΑ: 17, ΕΚΘΕΣΗ:16 ΦΥΣΙΚΗ:14
009	Παπαδόπουλος	Αγόρι, Ημ.γέννησης: 08/08/82	ΙΣΤΟΡΙΑ:16, ΕΚΘΕΣΗ:09, ΛΑΤΙΝΙΚΑ:12
007	Κονιτοπούλου	Κορίτσι, Ημ.γέννησης: 12/11/83	ΑΡΧΑΙΑ:19, ΝΕΑ:17
006	Βαλσάμη	Κορίτσι, Ημ.γέννησης: 10/05/82	ΜΑΘΗΜΑΤΙΚΑ:12, ΕΚΘΕΣΗ:15

Πίνακας 6.2. – Πρώτη εικόνα για τα διαθέσιμα στοιχεία των μαθητών του φροντιστηρίου.

Αν θεωρηθεί ότι τα πιο πάνω στοιχεία θα μπορούσαν να συνθέσουν μια οντότητα με το όνομα "ΜΑΘΗΤΗΣ", από άποψη σχεσιακής θεωρίας, η οντότητα αυτή και τα δεδομένα της είναι μη κανονικοποιημένα, ακριβώς γιατί περιέχουν επαναλαμβανόμενες πληροφορίες. Για να ξεκινήσει η κανονικοποίησή τους, πρέπει σαν πρώτο βήμα να εξαφανιστεί αυτού του είδους η επανάληψη δεδομένων, με τον τρόπο που προτείνεται στον **επόμενο πίνακα 6.3:**

Κωδικός μαθητή	Επίθετο	Φύλο	Ημερομηνία γέννησης	Μάθημα	Βαθμός
010	Πέτρου	Αγόρι	07/05/82	ΜΑΘΗΜΑΤΙΚΑ	17
010	Πέτρου	Αγόρι	07/05/82	ΕΚΘΕΣΗ	16
010	Πέτρου	Αγόρι	07/05/82	ΦΥΣΙΚΗ	14
009	Παπαδόπουλος	Αγόρι	08/08/82	ΙΣΤΟΡΙΑ	16
009	Παπαδόπουλος	Αγόρι	08/08/82	ΕΚΘΕΣΗ	09
009	Παπαδόπουλος	Αγόρι	08/08/82	ΛΑΤΙΝΙΚΑ	12
007	Κονιτοπούλου	Κορίτσι	12/11/83	ΑΡΧΑΙΑ	19
007	Κονιτοπούλου	Κορίτσι	12/11/83	ΝΕΑ	17
006	Βαλσάμη	Κορίτσι	10/05/82	ΜΑΘΗΜΑΤΙΚΑ	12
006	Βαλσάμη	Κορίτσι	10/05/82	ΕΚΘΕΣΗ	15

Πίνακας 6.3. - Δεδομένα σε πρώτη κανονική μορφή.

Τα δεδομένα, όπως παρουσιάζονται στον πιο πάνω πίνακα θεωρείται ότι βρίσκονται στην πρώτη κανονική μορφή *(first normal form - 1NF)*. Στη συγκεκριμένη μορφή, με βάση τη σχεσιακή θεωρία, βρίσκονται τα δεδομένα όταν δεν περιλαμβάνουν επαναλαμβανόμενες ομάδες δεδομένων.

Στο παράδειγμά μας παρατηρούμε ότι δεν είναι πλέον δυνατή η εμφάνιση κάτω από την ίδια ιδιότητα π.χ., του συνδυασμού "ΜΑΘΗΜΑΤΙΚΑ,ΕΚΘΕΣΗ", ενώ εμφανίζονται σαν ξεχωριστές ιδιότητες το "Φύλο" και η "Ημερομηνία γέννησης".

Παρόλα αυτά υπάρχουν πλειάδες, στις οποίες εμφανίζεται επανάληψη δεδομένων, π.χ., στις τρεις πρώτες πλειάδες επαναλαμβάνεται το επίθετο και το φύλο, κάτι βέβαια που συμβαίνει και σε επόμενες πλειάδες, γεγονός που ισχύει τόσες φορές για κάθε μαθητή, όσα τα μαθήματα που έχει επιλέξει.

Ένα άλλο πρόβλημα είναι ότι δεν υπάρχει εύκολος τρόπος για τη μεταβολή ή διαγραφή δεδομένων. Γιατί στην περίπτωση, για παράδειγμα, που ο μαθητής Πέτρου διαγραφεί, θα πρέπει να διαγραφούν τόσες πλειάδες, όσα και τα μαθήματα, τα οποία παρακολουθούσε. Επίσης, αν για κάποιο λόγο αποφασιστεί η αλλαγή του κωδικού κάποιου μαθητή, πρέπει να γίνει αλλαγή σε όλες τις πλειάδες που του αναλογούν. Όμως τέτοιου είδους αλλαγές μπορεί να οδηγήσουν σε προβλήματα σταθερότητας της βάσης και αξιοπιστίας των δεδομένων.

Απαιτείται, λοιπόν, να προχωρήσει σε επόμενα βήματα η διαδικασία της κανονικοποίησης. Η κανονικοποίηση εξαρτάται από το πρωτεύον κλειδί, το οποίο έχει ήδη επιλεγεί και το οποίο καθορίζει με μοναδικό τρόπο κάθε πλειάδα της οντότητας.

Αν έχει επιλεγεί για κλειδί ο κωδικός του μαθητή, προφανώς δεν μπορεί να χαρακτηρίσει με μοναδικό τρόπο μια πλειάδα, αφού, για παράδειγμα, ο κωδικός 010 χαρακτηρίζει τρεις από τις πλειάδες του παραδείγματος.

Επειδή σε κάθε μαθητή αντιστοιχούν τόσες πλειάδες, όσα και τα μαθήματα που παρακολουθεί, ο ιδανικός τρόπος για να προσδιοριστεί μονοσήμαντα μια πλειάδα είναι να χρησιμοποιηθεί σαν κλειδί προσδιορισμού ο συνδυασμός κωδικού και μαθήματος, δηλαδή έχουμε ένα σύνθετο πρωτεύον κλειδί. Επομένως, υπάρχει τώρα η δυνατότητα να προχωρήσουμε στον τυπικό ορισμό της οντότητας του παραδείγματος, με όρους της σχεσιακής θεωρίας:

ΜΑΘΗΤΗΣ(ΚΩΔΙΚΟΣ#,ΜΑΘΗΜΑ#,ΕΠΙΘΕΤΟ,ΦΥΛΟ,ΗΜ_ΓΕΝ,ΒΑΘΜΟΣ)

Το επόμενο βήμα στην κανονικοποίηση των δεδομένων είναι να διερευνηθεί αν υπάρχουν ιδιότητες, που δεν εξαρτώνται από το συνδυασμό τιμών των ιδιοτήτων που συνθέτουν το πρωτεύον κλειδί, αλλά μόνο από μία από τις δύο ιδιότητες. Πραγματικά, το επίθετο δεν εξαρτάται από το μάθημα, αλλά μόνο από τον κωδικό. Το ίδιο ισχύει για το φύλο και την ημερομηνία γέννησης. Στην περίπτωση αυτή επιβάλλεται να ακολουθηθεί η διαδικασία της διάσπασης *(decomposition)* της οντότητας ΜΑΘΗΤΗΣ σε δύο οντότητες ή σχέσεις, όπως θέλει να τις ονομάζει η σχεσιακή θεωρία:

Η μία οντότητα θα ονομαστεί "ΜΑΘΗΤΗΣ" και μπορεί με τυπικό τρόπο να οριστεί ως εξής:

ΜΑΘΗΤΗΣ(ΚΩΔΙΚΟΣ#,ΕΠΙΘΕΤΟ,ΦΥΛΟ,ΗΜ_ΓΕΝ)

Η δεύτερη οντότητα θα ονομαστεί "ΒΑΘΜΟΛΟΓΙΑ" και μπορεί να οριστεί ως εξής:
ΒΑΘΜΟΛΟΓΙΑ(ΜΑΘΗΜΑ#,ΚΩΔΙΚΟΣ_ΜΑΘΗΤΗ#,ΒΑΘΜΟΣ)

Η οντότητα (ή σχέση) "ΒΑΘΜΟΛΟΓΙΑ", έχει σαν πρωτεύον κλειδί το συνδυασμό των ιδιοτήτων "ΜΑΘΗΜΑ" και "ΚΩΔΙΚΟΣ_ΜΑΘΗΤΗ". Ο "ΚΩΔΙΚΟΣ_ΜΑΘΗΤΗ" είναι απαραίτητο να εισαχθεί στην οντότητα, έτσι ώστε μαζί με το "ΜΑΘΗΜΑ" να προσδιορίζει με μοναδικό τρόπο κάθε πλειάδα, αλλά και να αποτελέσει το **ξένο κλειδί,** που θα συνδέει, όπως η σχεσιακή θεωρία επιβάλλει, την οντότητα "ΒΑΘΜΟΛΟΓΙΑ" με την οντότητα "ΜΑΘΗΤΗΣ".

Τώρα τα δεδομένα βρίσκονται στη **δεύτερη κανονική μορφή,** όπου οι ιδιότητες σε κάθε οντότητα εξαρτώνται από ολόκληρο το πρωτεύον κλειδί και όχι από μέρος του. Μάλιστα, ταυτόχρονα βρίσκονται και στην λεγόμενη **τρίτη κανονική μορφή** (3NF), **αφού δεν υπάρχουν ιδιότητες, οι οποίες δεν εξαρτώνται από το πρωτεύον κλειδί.** Το γεγονός αυτό αποτελεί την προϋπόθεση για να θεωρηθεί ότι τα δεδομένα βρίσκονται στην τρίτη κανονική μορφή. Στις περισσότερες περιπτώσεις πραγματικών προβλημάτων, αλλά και στο παράδειγμά μας, όταν τα δεδομένα φτάσουν στη 2NF ικανοποιούν ταυτόχρονα και την 3NF.
Ο εμπνευστής της σχεσιακής θεωρίας E.F. Codd, καθόρισε και άλλες 2 κανονικές μορφές την

Η κανονικοποίηση πολλές φορές οδηγεί σε αργή προσπέλαση στα δεδομένα, ιδίως όταν υπάρχουν πολλές συσχετίσεις μεταξύ μεγάλων πινάκων. Σε τέτοιες περιπτώσεις και προκειμένου να εξυπηρετηθεί ο στόχος της ταχύτερης δυνατής πρόσβασης στα δεδομένα, μπορεί να μην ακολουθηθεί αυστηρά η θεωρία του Codd.

τέταρτη (4NF) και την πέμπτη (5NF) κανονική μορφή, οι οποίες στην πράξη τις περισσότερες φορές, δεν χρειάζεται να χρησιμοποιηθούν και για το λόγο αυτό δεν θα παρουσιαστούν στο παρόν βιβλίο. Όποιος, όμως, το επιθυμεί, μπορεί να αναζητήσει περισσότερες λεπτομέρειες στη βιβλιογραφία, που ακολουθεί στο τέλος του κεφαλαίου.

Συνοψίζοντας, η χρήση των κανόνων της σχεσιακής θεωρίας επιτρέπει την αντιμετώπιση προβλημάτων σωστής ενημέρωσης των δεδομένων, που προκύπτουν από την περιττή επανάληψη μέρους από αυτά στις πλειάδες. Ο αποκλεισμός αυτής της επανάληψης κάνει ευκολότερη την αποθήκευση των δεδομένων από ένα ΣΔΒΔ, όπως επίσης τη συντήρηση και τις μελλοντικές επεκτάσεις του συστήματος. Από την άλλη πλευρά, όταν το σύστημα αποτελείται από πολλές οντότητες συσχετισμένες μεταξύ τους με εξωτερικά κλειδιά, υπάρχει περίπτωση να επιβραδυνθεί η αναζήτηση στοιχείων από το ΣΔΒΔ. Για το λόγο αυτό σε κάποιες περιπτώσεις, όπου η γρήγορη ανάκτηση των δεδομένων είναι πρωτεύουσας σημασίας, μπορεί να μην ακολουθήσει κανείς αυστηρά τους κανόνες της κανονικοποίησης ή να τους προσαρμόσει στις ανάγκες του συγκεκριμένου συστήματος. Άρα **στην πράξη, ένα σχεσιακό σύστημα βάσης δεδομένων δεν ταυτίζεται αναγκαστικά με την απόλυτη εφαρμογή των κανόνων του Codd.**

Ξαναγυρίζοντας στο παράδειγμα, τα δεδομένα που αφορούν το μαθητή Πέτρου, χωρίζονται πλέον στις εξής πλειάδες ανά οντότητα **(πίνακες 6.4 & 6.5):**

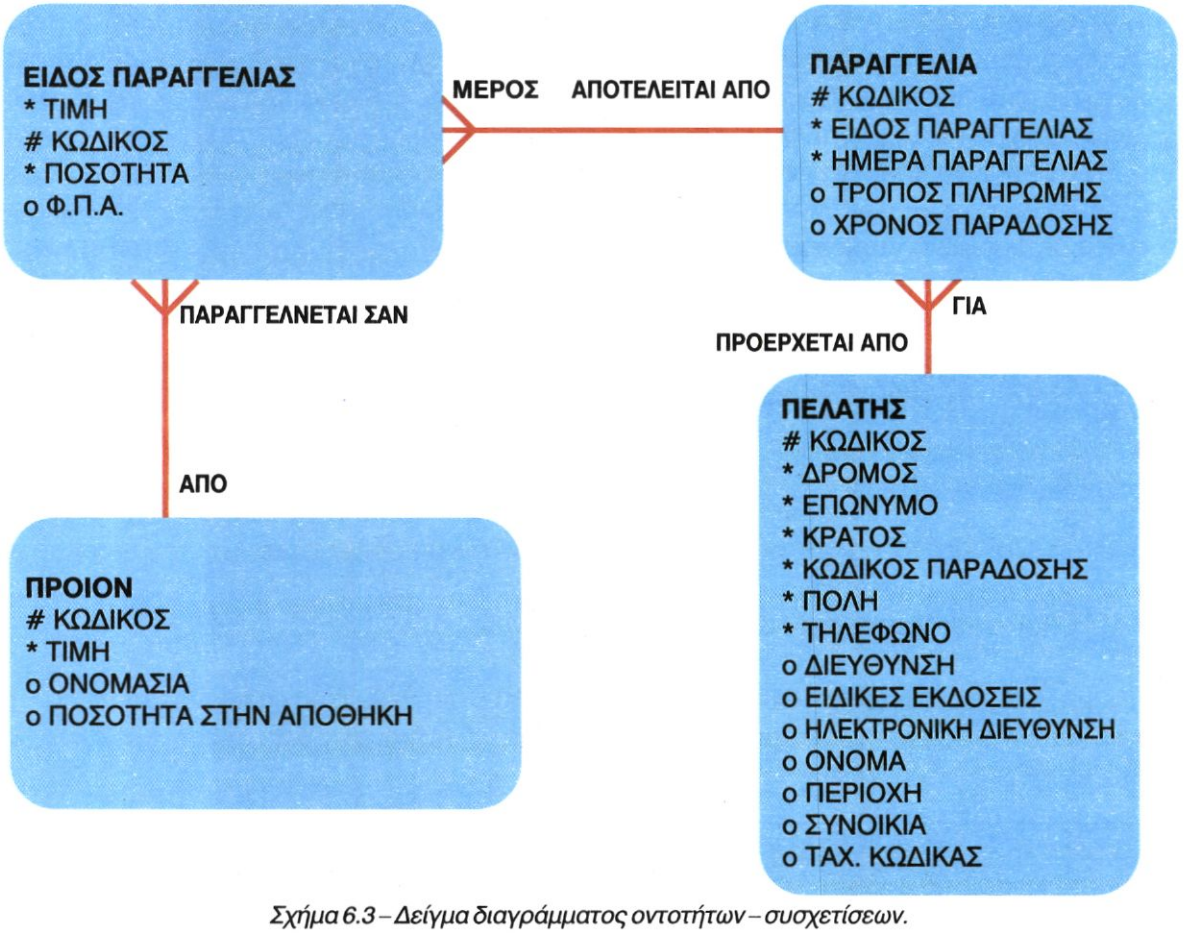
ΚΩΔΙΚΟΣ	ΕΠΙΘΕΤΟ	ΦΥΛΟ	ΗΜ_ΓΕΝ
010	Πέτρου	Αγόρι	07/05/82

Πίνακας 6.4. – Στοιχεία του μαθητή Πέτρου.

ΜΑΘΗΜΑ	ΚΩΔΙΚΟΣ_ΜΑΘΗΤΗ	ΒΑΘΜΟΣ
ΜΑΘΗΜΑΤΙΚΑ	010	17
ΕΚΘΕΣΗ	010	16
ΦΥΣΙΚΗ	010	14

Πίνακας 6.5. – Βαθμοί του μαθητή Πέτρου.

Μετά την ολοκλήρωση της διαδικασίας κανονικοποίησης, οι διαμορφωμένες πλέον οντότητες-σχέσεις απαιτείται να παρουσιαστούν με μορφή γλαφυρή και εύκολα αναγνώσιμη, ώστε όποιοι αναλάβουν την τελική υλοποίηση του συστήματος, να έχουν σαφή εικόνα για τα δεδομένα και τον τρόπο διασύνδεσής τους.



Σχήμα 6.3 – Δείγμα διαγράμματος οντοτήτων – συσχετίσεων.

Το εργαλείο που χρησιμοποιείται για την αποτύπωση της κατάστασης είναι ένας τύπος διαγράμματος, που ονομάζεται διάγραμμα οντοτήτων-συσχετίσεων *(entity-relationship diagram ή ERD diagram).* Στο διάγραμμα αυτό απεικονίζονται οι οντότητες που έχουν προκύψει από τη διαδικασία της κανονικοποίησης, οι σχέσεις μεταξύ τους και η κατηγορία της σχέσης π.χ., 1:N, M:N κ.ο.κ.

Ένα διάγραμμα οντοτήτων-συσχετίσεων δεν σχεδιάζεται πλέον με το χέρι, αλλά με ειδικό λογισμικό, που διευκολύνει την εποπτική παρουσίασή του, και απλοποιεί τη διαδικασία των αλλαγών και διορθώσεων, οι οποίες μπορεί να απαιτηθούν. Τα σύγχρονα ΣΔΒΔ συνήθως προσφέρονται με εργαλεία που βοηθούν και σ' αυτού του είδους τις εργασίες, ή προτείνουν ολοκληρωμένα πακέτα λογισμικού, για την υλοποίηση ενός συστήματος, από το στάδιο της ανάλυσης, μέχρι και τη δημιουργία του τελικού κώδικα. Τα πακέτα αυτά είναι γνωστά, σαν **εργαλεία βοηθούμενης από υπολογιστή σχεδίασης λογισμικού** *(computer aided software engineering tools ή case tools).*

Το διάγραμμα του **σχήματος 6.3** ένα τυπικό διάγραμμα οντοτήτων - συσχετίσεων, το οποίο σχεδιάστηκε με τη βοήθεια Η/Υ και απεικονίζει τις οντότητες και τις σχέσεις που προέκυψαν μετά από τη διαδικασία κανονικοποίησης των δεδομένων ενός συστήματος διαχείρισης παραγγελιών, που αναπτύσσεται για λογαριασμό αλυσίδας μαγαζιών παρασκευής πίτσας, με παραρτήματα σε διάφορες πόλεις. Στο διάγραμμα αυτό οι οντότητες απεικονίζονται με τα μπλε ορθογώνια, οι ιδιότητες που αποτελούν το πρωτεύον κλειδί κάθε οντότητας σημειώνονται με το σύμβολο (#), εκείνες που υποχρεωτικά πρέπει να έχουν κάποια τιμή συμβολίζονται με το(*) , ενώ εκείνες που προαιρετικά μόνο χρειάζεται να συμπληρωθούν με κάποια τιμή, συμβολίζονται με το (ο).

Οι σχέσεις μεταξύ των οντοτήτων απεικονίζονται με τα κόκκινα βέλη, τα οποία στη συγκεκριμένη περίπτωση στη μία άκρη τους παρουσιάζονται με τρεις μύτες, ενώ από την άλλη με μία. Ο συμβολισμός αυτός σημαίνει ότι η σχέση είναι πολλά προς ένα (Μ:1).

Για παράδειγμα, μία "ΠΑΡΑΓΓΕΛΙΑ" μπορεί να αποτελείται από πολλά είδη παραγγελλόμενων προϊόντων, π.χ., μία πίτσα και μία σαλάτα. Στις άκρες κάθε βέλους που συμβολίζει μια σχέση υπάρχει και ένα λεκτικό, που σε συντομία προσπαθεί να περιγράψει τη λογική που βρίσκεται πίσω από τη σχέση των οντοτήτων. Π.χ., το "ΕΙΔΟΣ" είναι "ΜΕΡΟΣ" μίας "ΠΑΡΑΓΓΕΛΙΑΣ", ενώ κάθε "ΠΑΡΑΓΓΕΛΙΑ" αποτελείται από τουλάχιστον ένα "ΕΙΔΟΣ".

Στην περίπτωση μίας σχέσης **πολλά προς πολλά (Μ:Ν)** ο συμβολισμός της σύνδεσης των σχετιζόμενων οντοτήτων, δεν είναι παρά ένα βέλος που καταλήγει **και στις δύο σχέσεις** με άκρη αποτελούμενη από τρεις μύτες.

Τονίζεται πάντως, πως η ανάγκη απεικόνισης σχέσεων πολλά προς πολλά σε διάγραμμα, αφορά στην ουσία **τα αρχικά στάδια** σχεδιασμού ενός κανονικοποιημένου συστήματος βάσεων δεδομένων, αφού στα τελευταία στάδια της κανονικοποίησης των οντοτήτων **καταλήγουμε πάντα σε σχέσεις πολλά προς ένα.**

Παράδειγμα:

Για να γίνει κατανοητή αυτή η αρχή δεν έχει κανείς παρά να ανατρέξει στο παράδειγμα που παρουσιάστηκε στα προηγούμενα με μαθητές και βαθμολογία. Στα πρώτα βήματα της οργάνωσης των σχετικών στοιχείων με βάση τις αρχές της κανονικοποίησης **(πίνακας 6.3),** υπήρχαν **πολλές** εγγραφές για τον ίδιο κωδικό μαθητή (π.χ., 010), για **πολλά** διαφορετικά μαθήματα (μία για κάθε μάθημα ξεχωριστά). Όμως στην τελική μορφή των οντοτήτων από τη μια είχαμε οντότητα με μοναδική καταχώρηση για κάθε μαθητή ξεχωριστά (οντότητα ΜΑΘΗΤΗΣ) και από την άλλη και από την άλλη την οντότητα "ΒΑΘΜΟΛΟΓΙΑ", με ξεχωριστή καταχώρηση για κάθε μάθημα και συσχέτιση με την οντότητα "ΜΑΘΗΤΗΣ", με κλειδί το "ΚΩΔΙΚΟΣ_ΜΑΘΗΤΗ". Η σχέση των δύο οντοτήτων είναι συμβατή τώρα με τους κανόνες κανονικοποίησης και η σχέση τους είναι: **ένας** (ΜΑΘΗΤΗΣ) **προς πολλά** (ΜΑΘΗΜΑΤΑ).
Με το σχεδιασμό των απαραίτητων διαγραμμάτων οντοτήτων συσχετίσεων, ολοκληρώνεται το πρώτο βήμα στη διαδικασία της ανάπτυξης ενός συστήματος βάσεων δεδομένων, από την πλευρά βέβαια της οργάνωσης των δεδομένων και μόνον και όχι της ανάπτυξης

των προγραμμάτων, η οποία εξελίσσεται σε συνδυασμό με την οργάνωση των δεδομένων. Το δεύτερο και τελικό βήμα είναι η μετάβαση από τις οντότητες σε πίνακες δεδομένων.

Σημείωση:

Στο 3ο μάθημα του τετραδίου εργασίας αρχίζουμε να χρησιμοποιούμε πολλές από τις εξειδικευμένες έννοιες των βάσεων που παρουσιάστηκαν πιο πάνω, εφαρμοσμένες στο γνωστό παράδειγμα (επισήμανση πρωτευόντων κλειδιών, πρώτη επισήμανση πιθανών συσχετίσεων κ.ο.κ).

6.3. ΠΙΝΑΚΕΣ ΚΑΙ ΣΥΣΧΕΤΙΣΕΙΣ ΠΙΝΑΚΩΝ

6.3.1. Περί πινάκων - γενικά

Η σχεσιακή θεωρία ορίζει μια βάση δεδομένων σαν συλλογή συσχετιζόμενων μεταξύ τους πινάκων. Το πρώτο στάδιο στη διαδικασία της διαμόρφωσης των πινάκων της βάσης είναι η επισήμανση οντοτήτων και η τελική μετατροπή τους σε πίνακες.

Στο πιο κάτω σχήμα 6.4 παρουσιάζεται η αντιστοιχία όρων μεταξύ οντότητας και πίνακα:

ΟΝΤΟΤΗΤΑ	⟷	ΠΙΝΑΚΑΣ (TABLE)
ΠΛΕΙΑΔΑ	⟷	ΓΡΑΜΜΗ (ROW)
ΙΔΙΟΤΗΤΑ	⟷	ΣΤΗΛΗ (COLUMN)

Σχήμα 6.4. - Αντιστοιχία οντότητας - πίνακα.

Ο τυπικός συμβολισμός ενός πίνακα ταυτίζεται στην πράξη με εκείνον μιας οντότητας:

ΟΝΟΜΑ_ΠΙΝΑΚΑ(ΙΔΙΟΤΗΤΑ1,ΙΔΙΟΤΗΤΑ2,ΙΔΙΟΤΗΤΑ3).

Εδώ, όμως, προκύπτει ένα ερώτημα: Γιατί άραγε χρειάζονται οι πίνακες, αφού υπάρχουν οι οντότητες; Μόνο και μόνο για τη διαφορετική τους ονοματολογία; Η απάντηση είναι ότι **οι οντότητες είναι πολύ πιο γενικές έννοιες από τους πίνακες και πέρα από το βασικό πλαίσιο του ορισμού τους και των σχέσεων που τις συνδέουν, ποτέ δεν περιλαμβάνονται στον ορισμό των οντοτήτων λεπτομέρειες, όπως ο ακριβής τύπος δεδομένων που χαρακτηρίζει μια ιδιότητα (π.χ., αν είναι τύπου χαρακτήρα ή ημερομηνίας ή αριθμητικός τύπος κ.ο.κ).** Επίσης, δεν υπάρχει καμία αναφορά για ειδικούς περιορισμούς που αφορούν στις οντότητες, στις σχέσεις που τις διέπουν και για ειδικούς ελέγχους ως προς τις επιτρεπόμενες τιμές των ιδιοτήτων τους. Στην ουσία, ο σχεδιασμός οντοτήτων και συσχετισμών υπάγεται στη διαδικασία της ανάλυσης, ενώ η μετατροπή των οντοτήτων σε πίνακες υπάγεται στη διαδικασία σχεδιασμού του συστήματος.

Η σχεσιακή θεωρία καθορίζει τους ακόλουθους κανόνες για τους πίνακες:

- Ο κάθε ξεχωριστός συνδυασμός γραμμής-στήλης έχει οπωσδήποτε μια ξεχωριστή τιμή. Ακό-

μα και όταν η λογική της συγκεκριμένης εφαρμογής επιτρέπει να μην υπάρχει τιμή σε κάποια στήλη κάποιας γραμμής, στην πραγματικότητα υπάρχει η ψευδοτιμή κενό *(null)*. Εξασφαλίζεται έτσι η προϋπόθεση ότι οι στήλες ενός πίνακα δεν αποτελούν επαναλαμβανόμενες ομάδες τιμών, που είναι και η βασική προϋπόθεση για να βρίσκονται τα δεδομένα στην πρώτη κανονική μορφή (1NF), έτσι ώστε να είναι απλούστερη η διαχείριση των δεδομένων.

- Κάθε γραμμή θεωρούμενη ως σύνολο τιμών πρέπει να είναι μοναδική. Απαγορεύεται να υπάρχουν δύο γραμμές με τις ίδιες ακριβώς τιμές. Υπάρχει μάλιστα τουλάχιστον μία στήλη ή συνδυασμός στηλών, οι τιμές των οποίων με μοναδικό τρόπο προσδιορίζουν κάθε γραμμή του πίνακα. Η στήλη ή οι στήλες αυτές αποτελούν αυτό που ονομάζεται πρωτεύον κλειδί, έννοια που συναντήσαμε και στις οντότητες. Ο κανόνας αυτός εγγυάται ότι κάθε γραμμή ενός πίνακα έχει ξεχωριστή σημασία, με ταυτότητά της την τιμή του πρωτεύοντος κλειδιού.
- Η σειρά των γραμμών σε ένα πίνακα δεν έχει σημασία. Πρόκειται για κανόνα ανάλογο με τον προηγούμενο, με τη διαφορά ότι αναφέρεται σε γραμμές και όχι σε στήλες και επιτρέπει την ανάκτηση των γραμμών ενός πίνακα σε οποιαδήποτε σειρά.
- Οι τιμές των στηλών είναι του ίδιου τύπου. Με όρους σχεσιακής θεωρίας, ο κανόνας αυτός σημαίνει ότι όλες οι τιμές σε μία συγκεκριμένη στήλη, πρέπει να προέρχονται από το ίδιο πεδίο ορισμού *(domain)*, δηλαδή από το ίδιο σύνολο επιτρεπόμενων τιμών. Ο κανόνας αυτός απλοποιεί την πρόσβαση στα δεδομένα επειδή, τόσο αυτοί που αναπτύσσουν το σύστημα, όσο και οι χρήστες του, μπορούν να είναι σίγουροι για τον τύπο των δεδομένων που περιλαμβάνονται σε μια στήλη. Επίσης απλοποιεί τη διαδικασία της επαλήθευσης και του ελέγχου των δεδομένων, αφού το σύστημα έχει να ελέγξει μεταξύ συγκεκριμένων τιμών αν η τιμή μιας στήλης περιλαμβάνεται σε αυτές.
- Η σειρά των στηλών του πίνακα δεν έχει καμία σημασία. Ο κανόνας αυτός σημαίνει ότι σε αντίθεση με τα αρχεία (όπου η σειρά των πεδίων μέσα στην εγγραφή έχει σημασία, τόσο στην προσθήκη νέων δεδομένων, όσο και στην αναζήτηση πληροφοριών από το αρχείο), στους πίνακες δεν παίζει κανένα ρόλο η σειρά με την οποία αναφέρεται κανείς στις στήλες μιας συγκεκριμένης εγγραφής του πίνακα, είτε για να προσθέσει, είτε για να αναζητήσει πληροφορίες. Επίσης, εξαιτίας αυτού του κανόνα, οποιαδήποτε αλλαγή στη φυσική δομή των δεδομένων μέσα στη βάση, δεν επηρεάζει τη λογική αναζήτησης πληροφοριών.
- Κάθε στήλη έχει την ξεχωριστή ονομασία της. Αφού δεν έχει σημασία η σειρά των στηλών μέσα στον πίνακα, λογικό είναι οι στήλες να αναφέρονται με το ξεχωριστό τους όνομα και όχι με τη σχετική τους θέση στον πίνακα. Όμως η μοναδικότητα του ονόματος μιας στήλης αφορά μόνο σε ένα συγκεκριμένο πίνακα και όχι σε όλους τους πίνακες της βάσης δεδομένων. Δηλαδή, είναι δυνατό μια στήλη με όνομα π.χ., "ΚΩΔΙΚΟΣ", να βρίσκεται σε πολλούς διαφορετικούς πίνακες της ίδιας βάσης.

Η **σχεσιακή θεωρία** ορίζει μια βάση δεδομένων ως συλλογή συσχετιζόμενων μεταξύ τους πινάκων. Ο συσχετισμός των πινάκων γίνεται με τη σύνδεση εξωτερικών κλειδιών μιας οντότητας με τα πρωτεύοντα κλειδιά άλλων οντοτήτων.

6.3.2. Συσχετίσεις πινάκων

Μία **σχέση ή συσχέτιση** *(relationship)* είναι η σύνδεση δύο ή περισσοτέρων πινάκων με τη λογική που ήδη παρουσιάστηκε μέσα από τις οντότητες, δηλαδή με τη σύνδεση εξωτερικών κλειδιών μιας οντότητας με τα πρωτεύοντα κλειδιά άλλων οντοτήτων.

Γενικά η ύπαρξη και η χρησιμότητα των κλειδιών, πρωτευόντων και εξωτερικών, είναι αποφασι-

στικής σημασίας για τις σχεσιακές βάσεις δεδομένων, επειδή επιτρέπουν τη συνδεσιμότητα μεταξύ πινάκων της βάσης, στο βαθμό που αυτό απαιτείται από τη λογική κάθε εφαρμογής. Το λογισμικό που χρησιμοποιείται για να προσπελάσει τη βάση και να ανακτήσει πληροφορίες, έχει τη δυνατότητα να συνδέσει δυναμικά γραμμές από περισσότερους του ενός συσχετιζόμενους πίνακες, ακολουθώντας τις συνδέσεις που παρέχουν τα κλειδιά.

Άλλο στοιχείο που χαρακτηρίζει τις βάσεις είναι η ακεραιότητα των δεδομένων *(data integrity)*. Ακεραιότητα των δεδομένων μιας βάσης σημαίνει ότι οι πίνακες της βάσης μπορούν να προσπελασθούν με ασφάλεια και σταθερότητα μέσα από τα προγράμματα εφαρμογών. Οι δύο βασικοί κανόνες που διασφαλίζουν την ακεραιότητα των δεδομένων είναι:

- **Ακεραιότητα οντοτήτων** *(entity integrity)*.
- **Ακεραιότητα συσχετίσεων** *(referential integrity)*.

Ακεραιότητα οντοτήτων σημαίνει ότι η τιμή του πρωτεύοντος κλειδιού σε ένα πίνακα, δεν μπορεί ποτέ να είναι η κενή (null) τιμή. Η κενή τιμή δε θα πρέπει να συγχέεται με το κενό διάστημα. Το κενό διάστημα αποτελεί δυνατή τιμή για μία στήλη που μπορεί να έχει εισαχθεί στη στήλη μέσα από κάποια διαδικασία εισαγωγής στοιχείων ή με κάποια εντολή ανάθεσης, ενώ η κενή τιμή δε χρειάζεται να εισαχθεί με ειδικό τρόπο σε μια στήλη. Αρκεί να μην έχει αποδοθεί στη στήλη αυτή τιμή με οποιαδήποτε διαδικασία. Επειδή το πρωτεύον κλειδί χρησιμοποιείται σαν ταυτότητα κάθε ξεχωριστής γραμμής ενός πίνακα, πρέπει πάντα να έχει καθορισμένη σαφώς, και ποτέ αόριστη ή άγνωστη τιμή. Οι διαδικασίες εισαγωγής, διαγραφής και ενημέρωσης ενός πίνακα βασίζονται στην ύπαρξη και μοναδικότητα του πρωτεύοντος κλειδιού, την οποία εξασφαλίζει ο κανόνας ακεραιότητας οντοτήτων.

Ακεραιότητα συσχετίσεων σημαίνει ότι, αν ένας πίνακας έχει ένα εξωτερικό κλειδί, που παραπέμπει σε κάποιον άλλον, η τιμή που αυτό έχει σε κάθε γραμμή του πίνακα πρέπει να ταιριάζει με κάποια τιμή του πρωτεύοντος κλειδιού στον πίνακα, στον οποίο αναφέρεται. Η ακεραιότητα συσχετίσεων περιλαμβάνει και ορισμένους **περιορισμούς** *(constraints)*, που καθορίζουν τους τρόπους με τους οποίους διασφαλίζεται αυτή η ακεραιότητα. Η λογική που διέπει την οργάνωση των δεδομένων προσδιορίζει ποιοι από αυτούς πρέπει να ενεργοποιηθούν κατά περίπτωση. Για να γίνει κατανοητό αυτό, ακολουθεί ένα παράδειγμα:

Όταν ένας πίνακας με τη χρήση εξωτερικού κλειδιού δείχνει σε κάποιον άλλο πίνακα, από τον οποίο έχει λογική εξάρτηση, τότε ο τελευταίος αυτός πίνακας αποτελεί για τον πρώτο τον **βασικό πίνακα** *(master table)* της σχέσης τους, ενώ ο άλλος θεωρείται **ο πίνακας λεπτομερειών** *(detail table)* ή αλλιώς **εξαρτώμενος πίνακας.**

Υπάρχουν, λοιπόν, περιορισμοί που, ανάλογα με τις απαιτήσεις, επιβάλουν όταν π.χ., γίνεται διαγραφή μιας γραμμής από το βασικό πίνακα, αυτόματα να διαγράφονται οι γραμμές από τον εξαρτώμενο πίνακα, στις οποίες το εξωτερικό κλειδί σύνδεσής τους, έχει σαν τιμή, την τιμή του

πρωτεύοντος κλειδιού της διαγραμμένης από το βασικό πίνακα γραμμής. Αυτού του είδους ο περιορισμός λέγεται περιορισμός εξαρτημένης διαγραφής *(cascade delete)*. Ανάλογοι περιορισμοί ισχύουν και στις ενημερώσεις ή διαγραφές των πινάκων και υπάρχουν για να αντιμετωπίσουν πραγματικά προβλήματα.

Αν το παράδειγμα που χρησιμοποιήθηκε σε προηγούμενες σελίδες και αναφερόταν στις οντότητες "ΜΑΘΗΤΗΣ" και "ΒΑΘΜΟΛΟΓΙΑ" επεκτεινόταν έτσι, ώστε από τις οντότητες αυτές να δημιουργηθούν οι αντίστοιχοι πίνακες, με την ίδια ακριβώς λογική, όπως το πιο κάτω **σχήμα 6.5** δείχνει, προκύπτουν τα εξής προβλήματα:

ΜΑΘΗΤΗΣ

ΚΩΔΙΚΟΣ	ΕΠΙΘΕΤΟ	ΦΥΛΟ	ΗΜ_ΓΕΝ
010	Πέτρου	Αγόρι	07/05/82

ΒΑΘΜΟΛΟΓΙΑ

ΜΑΘΗΜΑ	ΚΩΔΙΚΟΣ_ΜΑΘΗΤΗ	ΒΑΘΜΟΣ
ΜΑΘΗΜΑΤΙΚΑ	010	17
ΕΚΘΕΣΗ	010	16
ΦΥΣΙΚΗ	010	14

Σχήμα 6.5. - Στοιχεία από δύο συσχετιζόμενους πίνακες και κάποιες από τις σχετιζόμενες γραμμές τους.

1. Μπορεί μια γραμμή από το βασικό πίνακα, που εδώ είναι ο πίνακας "ΜΑΘΗΤΗΣ" (π.χ., η γραμμή που αντιστοιχεί στο μαθητή Πέτρου), να διαγραφεί, τη στιγμή που στον εξαρτημένο πίνακα "ΒΑΘΜΟΛΟΓΙΑ" έχουν εισαχθεί για το μαθητή αυτό βαθμοί στα μαθήματα της επιλογής του; Μήπως, για να γίνει διαγραφή γραμμής από το βασικό πίνακα, πρέπει πρώτα να μην υπάρχουν γραμμές στον εξαρτημένο πίνακα, που να έχουν σαν τιμή εξωτερικού κλειδιού την τιμή που έχει στο πρωτεύον κλειδί της η υπό διαγραφή γραμμή του βασικού πίνακα (εξαρτημένες γραμμές);

2. Μπορεί να αλλαχθεί στο βασικό πίνακα η τιμή της στήλης "ΚΩΔΙΚΟΣ" για το μαθητή Πέτρου, αφού αυτή ακριβώς η στήλη αποτελεί πρωτεύον κλειδί για τον πίνακα "ΜΑΘΗΤΗΣ"; Αν επιτραπεί κάτι τέτοιο, τότε όλες οι γραμμές από τον πίνακα "ΒΑΘΜΟΛΟΓΙΑ", που αναφέρονται στον Πέτρου, αρχικά θα δείχνουν σε μη υπαρκτό κλειδί, επομένως υπάρχει σοβαρό πρόβλημα ακεραιότητας δεδομένων. Οπότε πρέπει να υπάρξει κατάλληλος περιορισμός, ο οποίος, είτε δεν επιτρέπει τέτοιου είδους ενημέρωση, είτε φροντίζει αυτόματα για την ενημέρωση των εξαρτημένων γραμμών του άλλου πίνακα, όταν γίνει αλλαγή των περιεχομένων του πρωτεύοντος κλειδιού στο βασικό πίνακα.

Οι λύσεις στα προβλήματα 1 και 2 δίνονται από ειδικούς τύπους περιορισμών, τους οποίους επι-

τρέπει το ΣΔΒΔ και που ονομάζονται περιορισμοί διαγραφής (delete constraints), όπως είναι ο περιορισμός εξαρτημένης διαγραφής που είδαμε πιο πάνω στο πρόβλημα 1, ή περιορισμοί ενημέρωσης *(update constraints)*, που δίνουν λύσεις σε προβλήματα όπως το πρόβλημα 2.

Τα σύγχρονα ΣΔΒΔ δίνουν τη δυνατότητα επιβολής και άλλων τύπων περιορισμών, όπως είναι οι περιορισμοί ελέγχου *(check constraints)*, με τη χρήση των οποίων απαγορεύεται η εισαγωγή τιμών σε συγκεκριμένες στήλες, όταν δεν εκπληρώνουν τις προϋποθέσεις που θέτει αντίστοιχος περιορισμός. Π.χ., μπορεί να τεθεί ένας περιορισμός ελέγχου στη στήλη "ΒΑΘΜΟΣ" του πίνακα "ΒΑΘΜΟΛΟΓΙΑ", ο οποίος να επιβάλει οι τιμές που παίρνει η συγκεκριμένη στήλη να κυμαίνονται μεταξύ του 0 και του 20. Αν επιχειρηθεί στη στήλη αυτή η εισαγωγή τιμής έξω από τα όρια αυτά, τότε η εισαγωγή εμποδίζεται αυτόματα από το ΣΔΒΔ.

Τέλος, η σημαντικότερη ίσως εξασφάλιση των δεδομένων από τα σύγχρονα ΣΔΒΔ, είναι η **υποστήριξη συναλλαγών** *(transactions)*.

Ορισμός

Μια **συναλλαγή** είναι ένα σύνολο εντολών ενημέρωσης (εγγραφής ή διαγραφής), που μπορεί να αφορούν σε ένα ή περισσότερους πίνακες, αλλά θεωρούνται σαν πακέτο εντολών που πρέπει να εκτελεστεί ολόκληρο. Αν για οποιοδήποτε λόγο (λάθος εντολή ή βλάβη στο σύστημα) αποτύχει η εκτέλεση μιας από τις εντολές του πακέτου, τότε το ΣΔΒΔ απορρίπτει ολόκληρη τη συναλλαγή.

Η έννοια της ακεραιότητας μιας συναλλαγής είναι κορυφαίας σημασίας για την αξιοπιστία ενός ΣΔΒΔ, γιατί εξασφαλίζει την ακεραιότητα των δεδομένων της. Με τους περιορισμούς ακεραιότητας που εισάγουν σε τέτοιου είδους συναλλαγές, τα σύγχρονα ΣΔΒΔ θεωρούν ότι οριστικά έχει τελειώσει η συναλλαγή, τότε και μόνο τότε, όταν όλες οι εντολές που την αποτελούν έχουν τελειώσει οριστικά και με ασφαλή τρόπο. Σε κάθε άλλη περίπτωση ακυρώνονται τα αποτελέσματα όλων των εντολών που τυχόν πρόλαβαν να εκτελεστούν.

Μπορείτε να φανταστείτε τι θα συνέβαινε, αν σε μια τράπεζα που χρησιμοποιεί ένα ΣΔΒΔ για την ενημέρωση των λογαριασμών των πελατών της, το οποίο δεν διαθέτει υποστήριξη συναλλαγών, γινόταν διακοπή ρεύματος, ενόσω βρισκόταν σε εξέλιξη μια διαδικασία ενημέρωσης των αρχείων της;

Γενικά, ένα σύγχρονο σχεσιακό ΣΔΒΔ καλύπτει όλες τις απαιτήσεις ασφάλειας, ακεραιότητας των δεδομένων και αξιοπιστίας, που είναι αναγκαίες για να βασίσει επάνω τους ένα πληροφοριακό σύστημα την αποτελεσματική λειτουργία του και την ευελιξία της εισαγωγής τυχόν αναγκαίων μετατροπών στο μέλλον.

Σημείωση:

Το πιο ενδιαφέρον μέρος της εξάσκησης πάνω στο σχεδιασμό μιας απλής βάσης δεδομένων, καλύπτεται από το 4ο και 5ο μάθημα του τετραδίου εργασίας. Εδώ ενθαρρύνεται η εξάσκηση σε θέματα όπως η κατασκευή διαγραμμάτων οντοτήτων-συσχετίσεων και η κανονικοποίηση της βάσης. Πετυχαίνεται έτσι η σύνδεση των θεωρητικών γνώσεων που παρατέθηκαν στο βιβλίο αυτό με πρακτικές που θα μπορούσαν να βρουν εφαρμογή και σε πραγματικά προβλήματα σχεδιασμού βάσεων.

Ανακεφαλαίωση / περίληψη

Η οργάνωση των δεδομένων σε αρχεία συνιστάται σε εφαρμογές που δεν απαιτείται ιδιαίτερη πολυπλοκότητα στη σύνδεση των δεδομένων μεταξύ τους, ή, για διάφορους λόγους, έχει επιλεγεί σαν εργαλείο ανάπτυξης εφαρμογών μια γλώσσα προγραμματισμού, που δεν υποστηρίζει απευθείας οργάνωση των δεδομένων σε βάσεις. Μειονέκτημα του παραδοσιακού τρόπου οργάνωσης δεδομένων σε αρχεία είναι η αδυναμία αξιοποίησής τους με διαφορετικό από τον προβλεπόμενο από την εφαρμογή που τα χρησιμοποιεί τρόπο. Κάτι τέτοιο απαιτεί προγραμματιστική παρέμβαση στην εφαρμογή. Το μειονέκτημα αυτό ξεπερνούν τα ΣΔΒΔ.

Μια πολύ διαδεδομένη κατηγορία ΣΔΒΔ αποτελούν τα σχεσιακά ΣΔΒΔ, που βασίστηκαν στη θεωρία, που θεμελίωσε στις αρχές της δεκαετίας του '70 ο Edgar Codd. Τα δεδομένα οργανώνονται αρχικά με πιο γενικό τρόπο συνιστώντας οντότητες και σε τελικό στάδιο σε πίνακες. Η διαδικασία με την οποία οργανώνονται οι οντότητες μιας εφαρμογής, με βάση τη σχεσιακή θεωρία και τους κανόνες της ονομάζεται κανονικοποίηση και περιλαμβάνει διάφορες μορφές ομαδοποίησης των δεδομένων. Κυριότερες εξ αυτών είναι οι εξής:

- *Πρώτη κανονική μορφή (1NF): Κάθε στήλη ενός πίνακα περιέχει μόνο μία πληροφορία και ποτέ επαναλαμβανόμενες ομάδες πληροφοριών.*
- *Δεύτερη κανονική μορφή (2NF): Κάθε στήλη εξαρτάται από όλο το πρωτεύον κλειδί του πίνακα και όχι από μέρος του.*
- *Τρίτη κανονική μορφή (3NF): Δεν υπάρχει καμία στήλη οποιουδήποτε πίνακα που δεν εξαρτάται από το πρωτεύον κλειδί.*

Σε περισσότερο πολύπλοκες οργανώσεις δεδομένων είναι δυνατό να χρειαστεί η εφαρμογή και άλλων δύο κανόνων κανονικοποίησης, αν και αυτό δεν εφαρμόζεται συχνά στην πράξη.

Οι διασυνδέσεις των δεδομένων των πινάκων υλοποιούνται με τη χρήση εξωτερικών κλειδιών, τα οποία είναι στήλες ενός πίνακα, που αντιστοιχούν στο πρωτεύον κλειδί άλλων πινάκων.

Το ΣΔΒΔ που διαχειρίζεται τη βάση υποστηρίζει τη χρήση κανόνων και περιορισμών, που εξασφαλίζουν την ακεραιότητα και την ασφάλεια των δεδομένων. Οι περιορισμοί αυτοί μπορεί να αυτοματοποιούν εργασίες όπως διαγραφή, εισαγωγή, τροποποιήσεις δεδομένων κ.λ.π. κατά τρόπο που να μην επιτρέπουν εργασίες σε ένα βασικό πίνακα, χωρίς να ενημερωθεί κάποιος εξαρτημένος από αυτόν, έτσι ώστε οι σχετιζόμενοι πίνακες να βρίσκονται πάντα σε συντονισμό.

Τέλος, τα σύγχρονα ΣΔΒΔ, υποστηρίζουν αυτόνομες συναλλαγές, δηλαδή ένα σύνολο λογικά συνδεομένων μεταξύ τους εντολών προς τη βάση, που νοούνται σαν ενότητα. Με τον τρόπο αυτό διασφαλίζεται είτε η εκτέλεση, είτε η ακύρωση όλων. Έτσι επιτυγχάνεται η συνεκτικότητα διαδικασιών μαζικής ενημέρωσης της βάσης, καθώς και η συντονισμένη ενημέρωση των επιμέρους πινάκων.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Πώς οργανώνονται τα δεδομένα σε αρχεία; Τι σημαίνει πρακτικά η οργάνωση των αρχείων με δείκτες;
2. Παρακάτω ορίζονται με συντομία οι τρεις μορφές κανονικοποίησης. Να συμπληρωθούν τα κενά με τον αριθμό που αντιστοιχεί στη σωστή κανονική μορφή για κάθε ορισμό:
   -Δεν υπάρχει στήλη ενός πίνακα μη εξαρτώμενη από το πρωτεύον κλειδί (___NF).
   -Δεν υπάρχει στήλη ενός πίνακα που να περιέχει επαναλαμβανόμενες ομάδες πληροφοριών (___NF)
   -Κάθε στήλη οφείλει να εξαρτάται από ολόκληρο το πρωτεύον κλειδί και όχι από μέρος του μόνο (___NF).
3. Να κατασκευαστεί ένα διάγραμμα οντοτήτων - συσχετίσεων που να απεικονίζει τις οντότητες "ΜΑΘΗΤΗΣ", "ΒΑΘΜΟΛΟΓΙΑ", που χρησιμοποιήθηκαν σαν παραδείγματα στις προηγούμενες σελίδες και τη σχέση που τις συνδέει.
4. Στο πλαίσιο σχετικής συζήτησης στην τάξη να αναζητηθούν παραδείγματα από την καθημερινή ζωή ή το φυσικό περιβάλλον, στα οποία να μπορούν να επισημανθούν οντότητες συσχετιζόμενες μεταξύ τους σε απλές ενότητες δεδομένων και να επιχειρηθεί κανονικοποίησή τους μέχρι την 3η κανονική μορφή. Να γίνει ανταλλαγή απόψεων πάνω σε σχετικές προτάσεις των μαθητών.

Γλωσσάριο - Ευρετήριο όρων

αρχεία δεικτών *(index files)* - Βοηθητικά αρχεία που λειτουργούν σαν ευρετήρια, τα οποία περιέχουν κλειδιά και διευθύνσεις, που αντιστοιχούν σε εγγραφές ενός βασικού αρχείου, το οποίο δεν είναι κατ' ανάγκη ταξινομημένο. Τα αρχεία δεικτών αυξάνουν την ταχύτητα αναζήτησης και εντοπισμού δεδομένων στο αρχείο [119]

γεννήτρια αναφορών *(report generator)* - Πρόγραμμα που παράγει με εύκολο και γρήγορο τρόπο εκτυπώσιμες αναφορές, οι οποίες μπορούν να χρησιμοποιηθούν ανεξάρτητα ή να ενσωματωθούν σε ολοκληρωμένες εφαρμογές [124].

διαγράμματα οντοτήτων - συσχετίσεων *(entity - relationship diagram)* - Ειδικά διαγράμματα τα οποία απεικονίζουν με γλαφυρότητα τις οντότητες που θα χρησιμοποιηθούν από μία εφαρμογή, καθώς και τον τρόπο με τον οποίο συσχετίζονται [130].

εργαλεία βοηθούμενης από υπολογιστή σχεδίασης λογισμικού *(CASE Tools) (Computer Aided Software Engineering)* - Προγράμματα τα οποία βοηθούν στο σχεδιασμό και την υλοποίηση μιας εφαρμογής, ξεκινώντας από τα πρώτα στάδια και καταλήγοντας στην παραγωγή του μεγαλύτερου μέρους του κώδικά της [135].

κανονικοποίηση *(normalization)* - Διαδικασία με την οποία οργανώνονται οι οντότητες μιας εφαρμογής, με βάση τη σχεσιακή θεωρία και τους κανόνες της [130].

κατανεμημένες βάσεις δεδομένων *(distributed databases)* - Βάσεις δεδομένων, οι οποίες με τη βοήθεια της τεχνολογίας των σύγχρονων δικτύων, μοιράζουν τον όγκο των πληροφοριών τους σε διαφορετικά υπολογιστικά συστήματα, χωρίς οι χρήστες τους να αντιλαμβάνονται πού ακριβώς φυλάσσεται κάθε πληροφορία [126].

ξένο κλειδί *(foreign key)* - Μια στήλη ή συνδυασμός στηλών ενός πίνακα, που αντιστοιχούν σε στήλες ενός άλλου πίνακα, δημιουργώντας το συσχετισμό μεταξύ αυτών των πινάκων [128].
οντότητα *(entity)* - Κάθε συστατικό ενός υπό εξέταση συστήματος, για το οποίο μας ενδιαφέρει να συλλέξουμε πληροφορίες. Σε μία βάση δεδομένων αναφέρεται στο είδος της πληροφορίας που αποθηκεύεται, π.χ., ένα πρόσωπο, μία έννοια, ένα γεγονός Η οντότητα όταν χρησιμοποιείται κατά το σχεδιασμό μίας βάσης δεδομένων, αποτελεί τον πρόγονο ενός πίνακα [126].
πίνακας *(table)* - Στις βάσεις δεδομένων οι πληροφορίες δεν οργανώνονται σε αρχεία αποτελούμενα από εγγραφές, αλλά σε πίνακες αποτελούμενους από γραμμές και στήλες [139].
πλειάδα *(tuple)* - Το αντίστοιχο της εγγραφής στα αρχεία και της γραμμής στους πίνακες,όπως χρησιμοποιείται στις οντότητες [127].
πρωτεύον κλειδί *(primary key)* - Η στήλη ή ο συνδυασμός στηλών ενός πίνακα, που χαρακτηρίζουν με μοναδικό τρόπο κάθε γραμμή (row) του πίνακα [128].
συναλλαγή *(transaction)* - Ένα σύνολο εντολών, που εκτελούνται σαν πακέτο. Αν μία από όλες αποτύχει, αποτυγχάνει όλη η συναλλαγή. Η έννοια της συναλλαγής αναφέρεται κύρια στις βάσεις δεδομένων [141].
συστήματα διαχείρισης βάσεων δεδομένων ή ΣΔΒΔ *(data base management systems)* - Σύνολο προγραμμάτων που διαχειρίζονται την οργάνωση δεδομένων σε βάσεις, ανεξάρτητες από τις εφαρμογές των χρηστών που προσπελαύνουν τα δεδομένα [125].
σχεσιακό πρότυπο *(relational model)* - Πρότυπο που βασίζεται στις θεωρίες του Codd και εισηγείται την οργάνωση των δεδομένων σε βάσεις, οι οποίες αποτελούνται από πίνακες συσχετιζόμενους μεταξύ τους με χρήση των λεγόμενων εξωτερικών κλειδιών [127].

Βιβλιογραφία

Codd E.F., "Further Normalization of the Data Base Relational Model", Prentice-Hall, 1972.
Date C.J., "An Introduction to Database Systems", Addison-Wesley, 1981.
Hawryszkiewycz I.T., "Database Analysis and Design", Maxwell Macmillan International Editions, 1991.
Σκουρλά Χρ. Π., "Σχεσιακές Βάσεις Δεδομένων", Εκδόσεις Νέων Τεχνολογιών, 1992.

ΕΝΟΤΗΤΑ Γ'

ΣΧΕΔΙΑΖΩ - ΕΦΑΡΜΟΖΩ

ΚΕΦΑΛΑΙΟ 7

ΣΧΕΔΙΑΣΗ ΛΟΓΙΣΜΙΚΟΥ

ΠΕΡΙΕΧΟΜΕΝΑ

7.1. Τι είναι σχεδίαση
7.2. Ιεραρχικά διαγράμματα σχεδίασης λογισμικού
7.3. Σχεδίαση και ανάλυση "από πάνω προς τα κάτω"

ΕΙΣΑΓΩΓΗ

- Θα ήταν δυνατό να κατασκευάσουμε ένα κτίριο, χωρίς να υπάρχει αρχιτεκτονικό σχέδιο;

Ασφαλώς όχι, γιατί δεν θα γνωρίζαμε τον τρόπο δόμησης, τη διάταξη των ορόφων και των διαμερισμάτων, το που θα κατασκευαστεί ο ανελκυστήρας, το κλιμακοστάσιο ή που θα είναι η είσοδος του κτιρίου κ.λ.π. Ακόμα και ο πιο έμπειρος κατασκευαστής δεν θα μπορούσε να κάνει τίποτα χωρίς τα κατάλληλα σχέδια.

Το **λογισμικό** (*software*) που χρησιμοποιούν οι ηλεκτρονικοί υπολογιστές, στις περισσότερες περιπτώσεις, είναι μια εξαιρετικά πολύπλοκη κατασκευή. Όπως κάθε πολύπλοκη κατασκευή, έτσι και το λογισμικό, πριν αρχίσει να κατασκευάζεται (να αναπτύσσεται) απαιτείται να υπάρχει κάποιο σχέδιο κατασκευής. Το στάδιο της σχεδίασης στη διαδικασία ανάπτυξης μιας εφαρμογής λογισμικού είναι εντελώς απαραίτητο και σε καμία περίπτωση δεν θα μπορούσε να παραληφθεί.

- Όμως, σε ποια ερωτήματα απαντά και τι περιλαμβάνει η σχεδίαση;
- Ποια εργαλεία υπάρχουν και ποιες τεχνικές χρησιμοποιούνται στη σχεδίαση του λογισμικού;
- Ποια είναι η μορφή και ποια η χρησιμότητα των παραγόμενων σχεδίων;

Μέσα από το απλό παράδειγμα που παρουσιάζεται στο κεφάλαιο αυτό δίνονται απαντήσεις στα παραπάνω ερωτήματα, τόσο σε θεωρητικό επίπεδο, όσο και σε επίπεδο πρακτικής.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να σχεδιάζει τμηματικά τη λύση ενός προβλήματος, με τη βοήθεια Η/Υ. Κάθε τμήμα του σχεδίου θα πρέπει να αναπτύσσεται από μια ομάδα μαθητών. Τα εργαλεία που θα χρησιμοποιεί η κάθε ομάδα για το σχεδιασμό της λύσης θα είναι τα ιεραρχικά διαγράμματα ροής.
2. Να συνθέτει τις εργασίες των επιμέρους ομάδων σε ένα τελικό διάγραμμα.
3. Να εκτελεί τους ελέγχους που θα επικυρώνουν το σχέδιο της λύσης που θα υλοποιηθεί.

7.1. ΤΙ ΕΙΝΑΙ Η ΣΧΕΔΙΑΣΗ

Ορισμός

Σχεδίαση (*design*) είναι μια διαδικασία κατά την οποία δημιουργούμε ένα ομοίωμα του κατασκευάσματος, το οποίο ονομάζεται σχέδιο.

Σαν πρώτη ύλη χρησιμοποιούμε χαρτί, μολύβι και μια ειδική γλώσσα που λέγεται γλώσσα σχεδίασης. Επίσης σε πολλές περιπτώσεις χρησιμοποιούμε ηλεκτρονικό υπολογιστή και ειδικά εργαλεία σχεδίασης, όπως π.χ., ειδικά προγράμματα, που μας βοηθούν να σχεδιάσουμε τα δικά μας προγράμματα.

Η σχεδίαση μας επιτρέπει να πειραματιστούμε με φθηνά υλικά (χαρτί και μολύβι), να γυρίσουμε πίσω και να διορθώσουμε ή να αναθεωρήσουμε το σχέδιο και να αποκτήσουμε γνώση που θα χρησιμοποιήσουμε στην κατασκευή. Τα ερωτήματα που πρέπει να απαντήσουμε στη φάση αυτή είναι τα εξής:

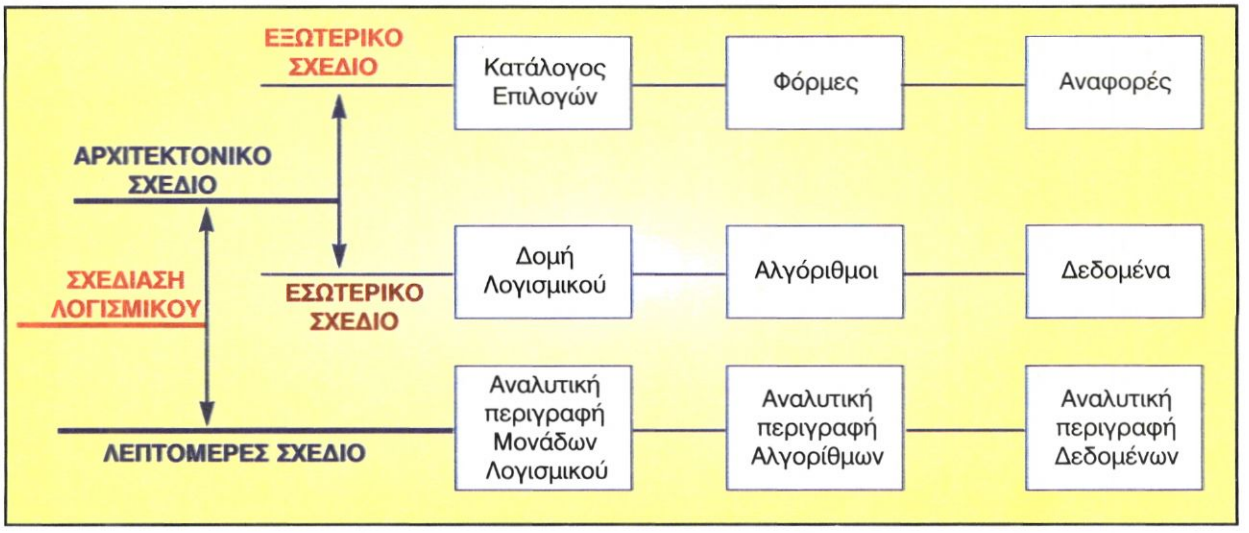
- ***Ποια είναι η δομή του συστήματος (αναλυτική περιγραφή);***
- ***Ποιες διαδικασίες απαιτούνται;***
- ***Πως θα γίνουν οι δοκιμές ελέγχου;***

Εκτός από την κατασκευή, το σχέδιο μας βοηθάει και στο να απαντηθούν γενικότερα ερωτήματα, που προκύπτουν καθ' όλη τη διάρκεια του κύκλου ανάπτυξης της εφαρμογής. Χαρακτηριστικά ερωτήματα αυτής της κατηγορίας είναι τα εξής:

- ***Πως θα γίνει ο χρονοπρογραμματισμός των εργασιών, δηλαδή ποιες εργασίες προηγούνται και ποιες έπονται;***
- ***Ποια θα είναι η χρονική διάρκεια των εργασιών, άρα και του έργου συνολικά;***
- ***Πως θα γίνει η ανάθεση εργασιών σε επί μέρους ομάδες (ποιος θα κάνει τι και πότε);***
- ***Πως θα γίνει η σύνθεση των εργασιών των επί μέρους ομάδων;***

Το σχέδιο, που είναι το προϊόν της φάσης της σχεδίασης, χρησιμοποιείται, εκτός από τη φάση της υλοποίησης και στις φάσεις της χρήσης και της συντήρησης της εφαρμογής. Θέματα που έχουν αναπτυχθεί κατά τη φάση της σχεδίασης εμπεριέχονται στα εγχειρίδια χρήσης. Έτσι, για παράδειγμα, το σχέδιο του πίνακα οργάνων ενός αυτοκινήτου χρησιμοποιείται για να βοηθήσει τον οδηγό στη χρήση των οργάνων, όπως επίσης και το ηλεκτρολογικό σχέδιο του αυτοκινήτου χρησιμοποιείται από το συντηρητή του για την προκαθορισμένη συντήρηση ή την αποκατάσταση βλαβών.

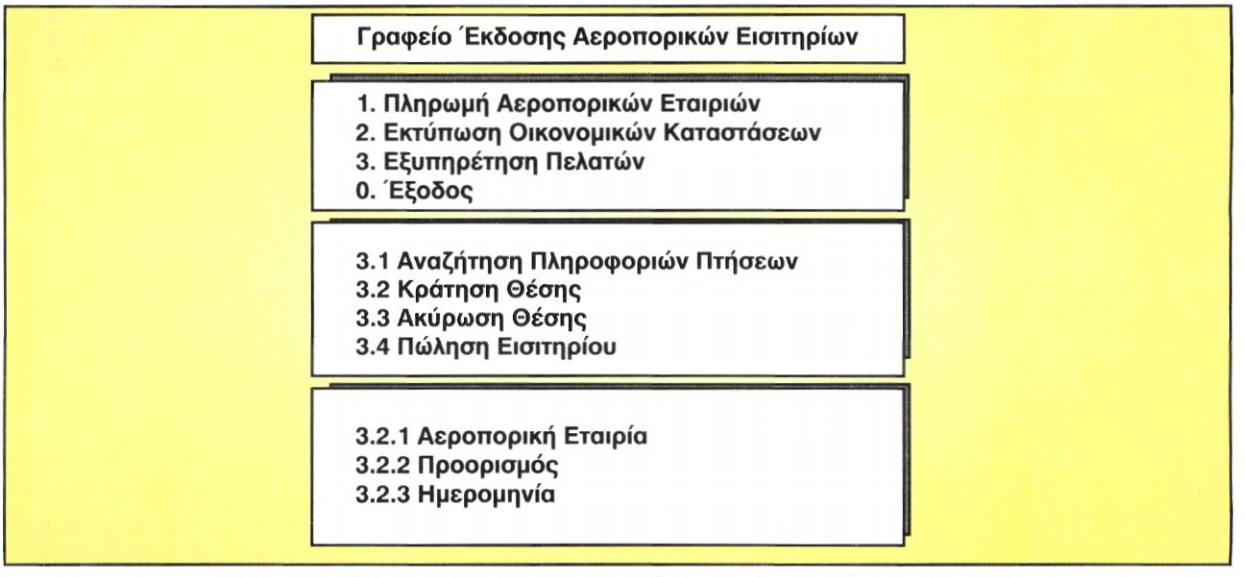
Η σχεδίαση περιλαμβάνει τη δημιουργία του **αρχιτεκτονικού** (*architectural*) ή **προκαταρκτικού** σχεδίου και του **λεπτομερούς** (*detailed*) σχεδίου. Ο αρχιτεκτονικός σχεδιασμός του συστήματος περιλαμβάνει τη δημιουργία του **εξωτερικού** *(external)* σχεδίου και του **εσωτερικού** *(internal)* σχεδίου (βλέπε **σχήμα 7.1**).



Σχήμα 7.1. - Σχεδίαση του Λογισμικού.

Το εξωτερικό σχέδιο παρουσιάζει την εξωτερική αρχιτεκτονική του συστήματος και περιγράφει τις λειτουργίες του συστήματος. Η κατασκευή του ξεκινά κατά τη φάση της ανάλυσης των απαιτήσεων και συνεχίζεται κατά τη φάση της σχεδίασης. Πρακτικά δεν είναι δυνατό να γίνει ο καθορισμός των απαιτήσεων χωρίς να γίνει ένα προκαταρκτικό σχέδιο, που να παρουσιάζει μια υψηλού επιπέδου άποψη της δομής του συστήματος και των λειτουργιών του. Το εξωτερικό σχέδιο περιλαμβάνει τις **διεπαφές** *(interfaces)* με το χρήστη (π.χ., **κατάλογοι επιλογών** των λειτουργιών *(menus))*, τη μορφή *(format)* των **αναφορών** *(reports)* κ.λ.π.

Παράδειγμα: Στο παρακάτω σχήμα 7.2. δίνεται ένα παράδειγμα εξωτερικού σχεδίου μηχανογραφικού συστήματος γραφείου έκδοσης αεροπορικών εισιτηρίων.



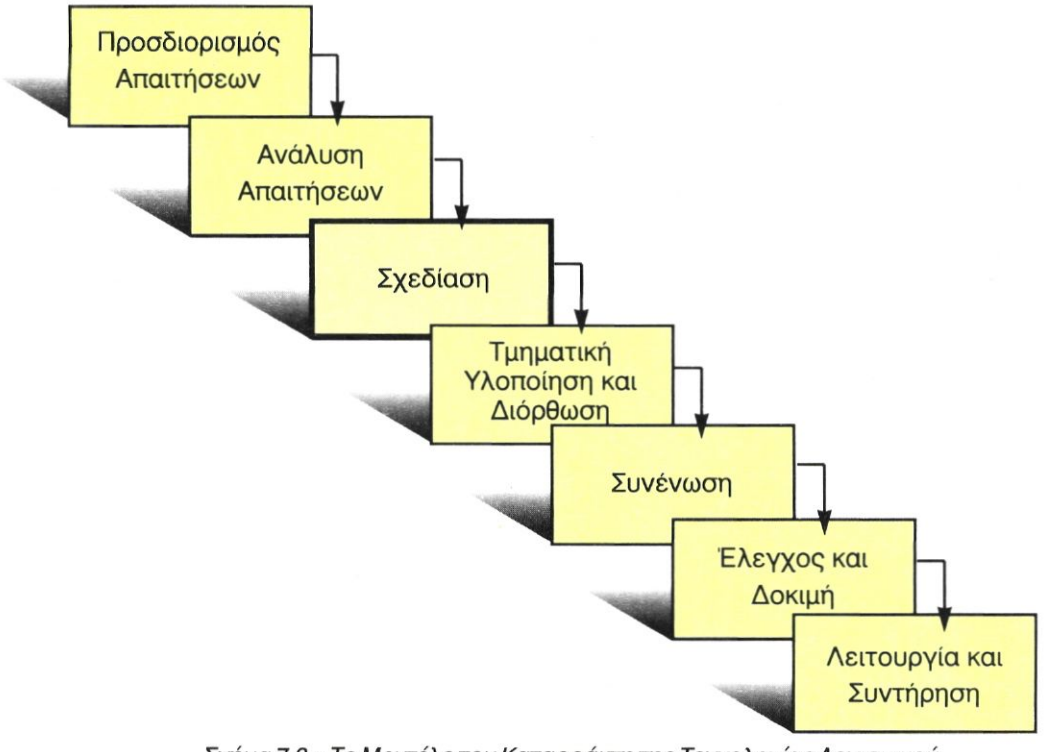
Σχήμα 7.2. - Εξωτερικό Σχέδιο Λογισμικού: Διεπαφή με τον Χρήστη (menus).

Το **εξωτερικό σχέδιο** πρέπει να παρουσιάζει στους χρήστες τι θα κάνει το σύστημα σε κάθε δική τους ενέργεια. Πρέπει να ενσωματώνει όλες τις υπάρχουσες απαιτήσεις, να είναι ανεξάρτητο της υλοποίησης και να είναι σχεδιασμένο με τρόπο που να είναι κατανοητός στο χρήστη ή γενικότερα σε αυτόν που ενδιαφέρεται να μελετήσει την εφαρμογή. Π.χ., μερικές από τις λειτουργίες - διαδικασίες που θα πρέπει ν' αναπτυχθούν σ' ένα σύστημα έκδοσης αεροπορικών εισιτηρίων, φαίνονται από τους καταλόγους επιλογών *(menu)* του **σχήματος 7.2**.

Το **εσωτερικό σχέδιο** απευθύνεται σε αυτούς που θα υλοποιήσουν την εφαρμογή (προγραμματιστές). Παρουσιάζει την εσωτερική αρχιτεκτονική του συστήματος, τη δομή του λογισμικού, τους αλγόριθμους, τα δεδομένα, τις ροές και τις δομές των δεδομένων και περιγράφει πως θα υλοποιηθούν οι λειτουργίες του με κάθε λεπτομέρεια. Δηλαδή, περιλαμβάνει όλες τις τεχνικές λεπτομέρειες και τον τρόπο υλοποίησης του προς ανάπτυξη συστήματος.

Το αρχιτεκτονικό σχέδιο απεικονίζει τις μονάδες (δηλ. δομικά στοιχεία που συμβολίζουν τις λειτουργίες της εφαρμογής και οι οποίες εξηγούνται αναλυτικά παρακάτω), χωρίς να παρέχει λεπτομερείς πληροφορίες για το πώς αυτές θα υλοποιηθούν. Αυτό γίνεται στη **λεπτομερή σχεδίαση** του λογισμικού. Στο λεπτομερές σχέδιο περιγράφεται ο αλγόριθμός που χρησιμοποιεί κάθε μονάδα, οι δομές δεδομένων, οι τύποι δεδομένων καθώς και οι σχέσεις μεταξύ λειτουργιών και δεδομένων.

Η σχεδίαση του λογισμικού είναι μια διαδικασία η οποία χρονικά έπεται των φάσεων του προσδιορισμού των απαιτήσεων και τις ανάλυσης των απαιτήσεων όπως φαίνεται και στο **σχήμα 7.3** που ακολουθεί:



Σχήμα 7.3. - Το Μοντέλο του Καταρράκτη της Τεχνολογίας Λογισμικού.

7.2. ΙΕΡΑΡΧΙΚΑ ΔΙΑΓΡΑΜΜΑΤΑ ΣΧΕΔΙΑΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Οι διαγραμματικές τεχνικές που χρησιμοποιούμε για την αποτύπωση του σχεδίου χωρίζονται σε δύο βασικές κατηγορίες:

1) Διαγραμματικές τεχνικές που χρησιμοποιούνται για την **γενική περιγραφή** του λογισμικού. Αυτές χρησιμοποιούν :

➢ Διαγράμματα ροής δεδομένων (*data flow diagrams).*

➢ Διαγράμματα δομής (*structure charts*).

➢ Διαγράμματα HIPO (*HIPO diagrams*).

2) Διαγραμματικές τεχνικές που χρησιμοποιούνται για τη **λεπτομερειακή περιγραφή** του λογισμικού. Αυτές χρησιμοποιούν :

➢ Διαγράμματα HIPO.

➢ Ψευδοκώδικα (*pseudocode*).

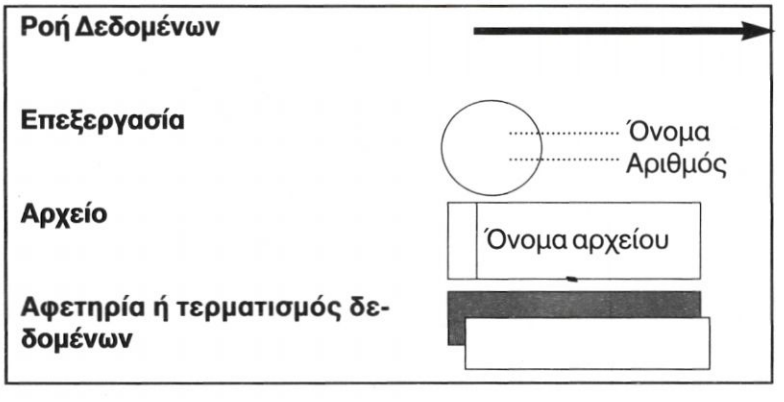
Τα παραπάνω διαγράμματα, που εξετάζονται αναλυτικά στη συνέχεια, συχνά αναφέρονται και σαν ιεραρχικά διαγράμματα, γιατί έχουν τη μορφή δέντρου. Για τον ίδιο λόγο και οι τεχνικές αυτές ονομάζονται και **δομημένες διαγραμματικές τεχνικές**.

7.2.1. Διαγράμματα Ροής Δεδομένων

Τα Διαγράμματα Ροής Δεδομένων (ΔΡΔ) χρησιμοποιούνται για να απεικονίσουν τη ροή των δεδομένων σε ένα σύστημα και επομένως αποτελούν ένα τρόπο παρουσίασης του συστήματος με τη μορφή δικτύου. Βασικά στοιχεία ενός διαγράμματος ροής δεδομένων είναι τα εξής:

- Η ροή δεδομένων που είναι ένα κανάλι από όπου "ρέουν" δεδομένα.
- Η επεξεργασία η οποία μετασχηματίζει δεδομένα.
- Το αρχείο στο οποία αποθηκεύουμε δεδομένα.
- Οι πηγές άντλησης των δεδομένων και τα σημεία τερματισμού τους.

Ο συμβολισμός των στοιχείων ενός ΔΡΔ φαίνεται στον πίνακα που ακολουθεί (**πίνακας 7.1**.):



Πίνακας 7.1. - Στοιχεία ενός Διαγράμματος Ροής Δεδομένων

Μια **ροή δεδομένων** (*data flow*) είναι ένα κανάλι μέσα από το οποίο περνούν τα δεδομένα και έχει τους εξής περιορισμούς:

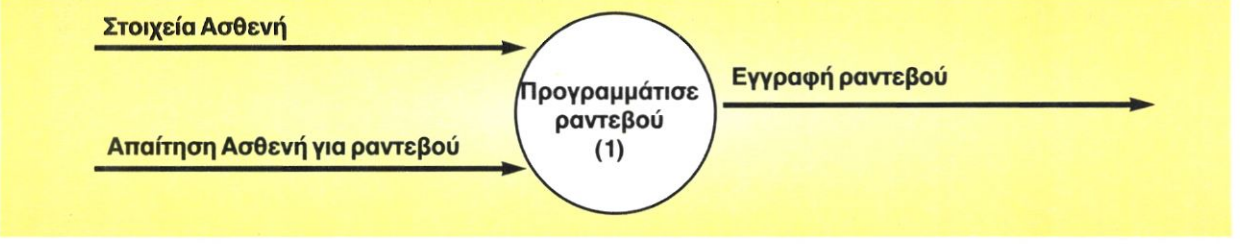
- Δεν συνδέει απευθείας μια αφετηρία ή ένα τερματισμό με ένα αρχείο.
- Έχει υποχρεωτικά ένα όνομα ως ετικέτα. Το όνομα πρέπει να είναι ουσιαστικό και όχι ρήμα.

Με τον όρο **επεξεργασία** (*process*) εννοούμε το τμήμα του συστήματος το οποίο μετασχηματίζει δεδομένα εισόδου (inputs) (αναφέρονται και σαν δεδομένα εισόδου) σε εξερχόμενα δεδομένα εξόδου (outputs) (αναφέρονται και σαν δεδομένα εξόδου). Μια επεξεργασία έχει τα εξής χαρακτηριστικά:

- Έχει ένα ***όνομα*** το οποίο δεν θα πρέπει να είναι επίθετο ή επίρρημα.
- Περιγράφει ένα απλό ***μετασχηματισμό*** (*transform*) δεδομένων που απαιτεί λίγους υπολογισμούς ή μπορεί να αναφέρεται σε μετασχηματισμούς που απαιτούν πολύπλοκους υπολογισμούς (ανάλογα με το επίπεδο ανάλυσης).
- Συνδέεται με μια ***αφορμή εκκίνησης*** (*trigger*) που ενεργοποιεί την επεξεργασία.
- Μια επεξεργασία απορροφά τα ***δεδομένα*** που χρειάζεται όταν συμβαίνει μια αφορμή εκκίνησης. Δε διέρχονται ποτέ δεδομένα χωρίς να υποστούν μια επεξεργασία.

Παράδειγμα:

Έτσι για παράδειγμα η επεξεργασία "Προγραμμάτισε ραντεβού" για το σύστημα "Ιατρείο" θα έχει ως δεδομένα εισόδου τα "Στοιχεία ασθενή" και "Απαίτηση ασθενή για ραντεβού" και ένα κανάλι εξόδου "Εγγραφή ραντεβού". Η απαίτηση του ασθενή είναι η αφορμή εκκίνησης έτσι ώστε με βάση τα στοιχεία του ασθενή να έχουμε την εγγραφή για το ραντεβού (**σχήμα 7.4**).

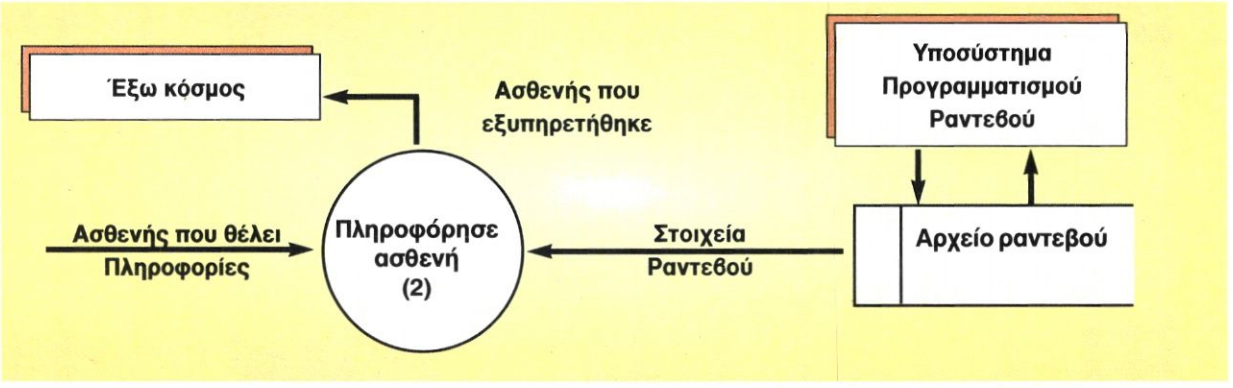


Σχήμα 7.4. - Επεξεργασία σε ΔΡΔ.

Τα **αρχεία** (*files*) είναι χώροι αποθήκευσης δεδομένων. Σε ένα αρχείο ενός ΔΡΔ θα πρέπει να καταλήγει και να φεύγει τουλάχιστον από μια ροή δεδομένων αντίστοιχα. Το όνομα ενός αρχείου θα πρέπει να είναι ουσιαστικό. Για παράδειγμα η επεξεργασία (1) στο σχήμα 7.4. θα μπορούσε να χρησιμοποιεί ένα αρχείο στο οποίο θα αποθηκεύονται τα στοιχεία των ραντεβού (**σχήμα 7.5**).

Σχήμα 7.5. - Χρήση Αρχείου σε ΔΡΔ.

Οι **αφετηρίες** και οι **καταλήξεις** είναι οντότητες που δέχονται ή παρέχουν δεδομένα. Οι οντότητες μπορεί να είναι φυσικά πρόσωπα, ή άλλα συστήματα που δεν ανήκουν στο υπό μελέτη σύστημα. Μπορεί όμως να είναι και στοιχεία του συστήματος. Μια αφετηρία ή μια κατάληξη επικοινωνεί με ένα αρχείο μέσω κάποιας επεξεργασίας. Έχουν ένα όνομα που είναι ουσιαστικό και αντιστοιχεί στην φυσική οντότητα καθώς επίσης και έναν κωδικό. Στο σχήμα που ακολουθεί δίνεται ένα παράδειγμα χρήσης κατάληξης και αφετηρίας (**σχήμα 7.6**).



Σχήμα 7.6. - Χρήση Κατάληξης και Αφετηρίας σε ΔΡΔ.

7.2.2 Διαγράμματα Δομής.

Ποιά είναι τα δομικά συστατικά ενός Διαγράμματος Δομής;

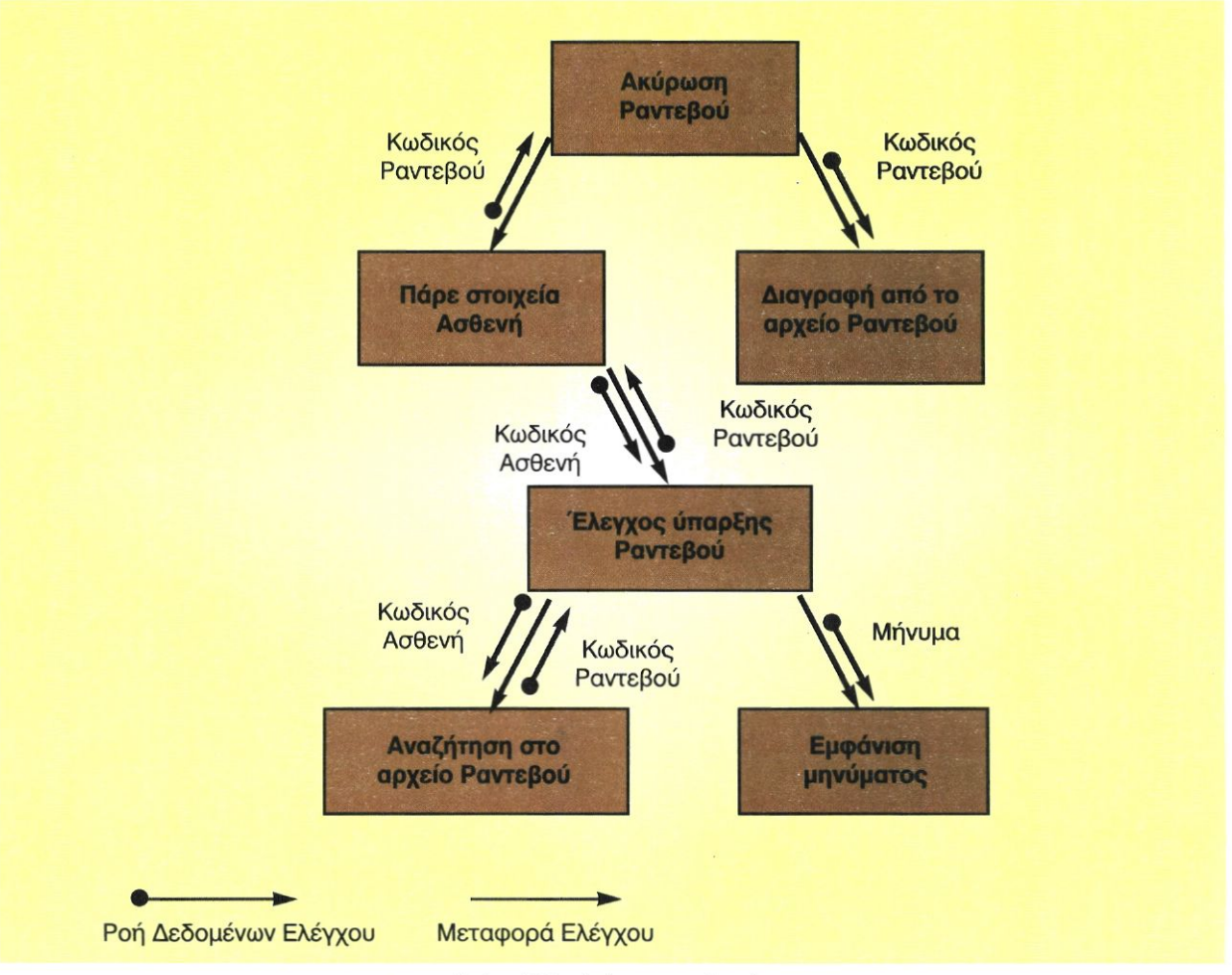
Τα **διαγράμματα δομής** *(structure charts)* χρησιμοποιούνται για την αποτύπωση του αρχιτεκτονικού σχεδίου του λογισμικού. Βασικό δομικό συστατικό τους είναι η **μονάδα** *(module)*. Οι μονάδες απεικονίζουν τις ενέργειες που κάνει το πρόγραμμα, όταν δέχεται μια εντολή του χρήστη και συμβολίζονται με ορθογώνια παραλληλόγραμμα. Η μεταφορά του ελέγχου του προγράμματος από μία μονάδα σε άλλη συμβολίζεται με βέλη που ενώνουν τα παραλληλόγραμμα μεταξύ τους. Ένα βέλος, που ενώνει δυο μονάδες δείχνει ότι κατά την εκτέλεση του προγράμματος ο έλεγχος περνάει από τη μία μονάδα στην άλλη με τη φορά του βέλους. Μία μονάδα μπορεί να καλεί περισσότερες από μία μονάδες, ενώ είναι δυνατό περισσότερες από μία μονάδες να μεταφέρουν τον έλεγχο στην ίδια μονάδα.

Τα διαγράμματα δομής έχουν ιεραρχική (δενδροειδή) μορφή. Στην κορυφή ενός διαγράμματος δομής υπάρχει μια μονάδα που ονομάζεται ρίζα. Από τη ρίζα ο έλεγχος μεταφέρεται στις μονάδες των πιο κάτω επιπέδων, αλλά πάντα επιστρέφει στη μονάδα που καλεί τις υπόλοιπες. Ανάμεσα σε δύο μονάδες υπάρχει το πολύ μία σχέση. Έτσι, αν η μονάδα Α καλέσει τη μονάδα Β, τότε η Β δεν μπορεί να καλέσει την Α.

Όταν μεταφέρεται ο έλεγχος σε μια μονάδα, μεταφέρονται συνήθως και δεδομένα. Τα δεδομένα μεταφέρονται προς οποιαδήποτε κατεύθυνση, ανεξάρτητα από τη φορά με την οποία μεταφέρεται ο έλεγχος. Τα ονόματα των δεδομένων γράφονται δίπλα στο βέλος μεταφοράς του ελέγχου.

Παράδειγμα:

Στο **σχήμα 7.7** παρουσιάζεται ένα διάγραμμα δομής για την ακύρωση ενός ραντεβού στο σύστημα "Ιατρείο".

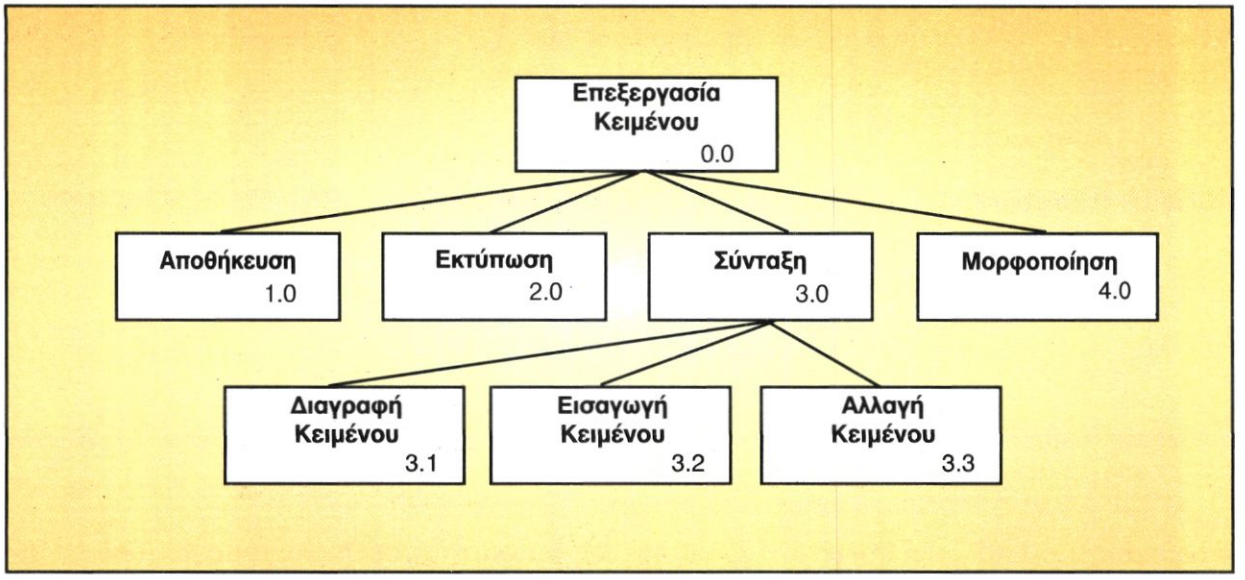


Σχήμα 7.7. - Διάγραμμα Δομής.

7.2.3. Διαγράμματα HIPO

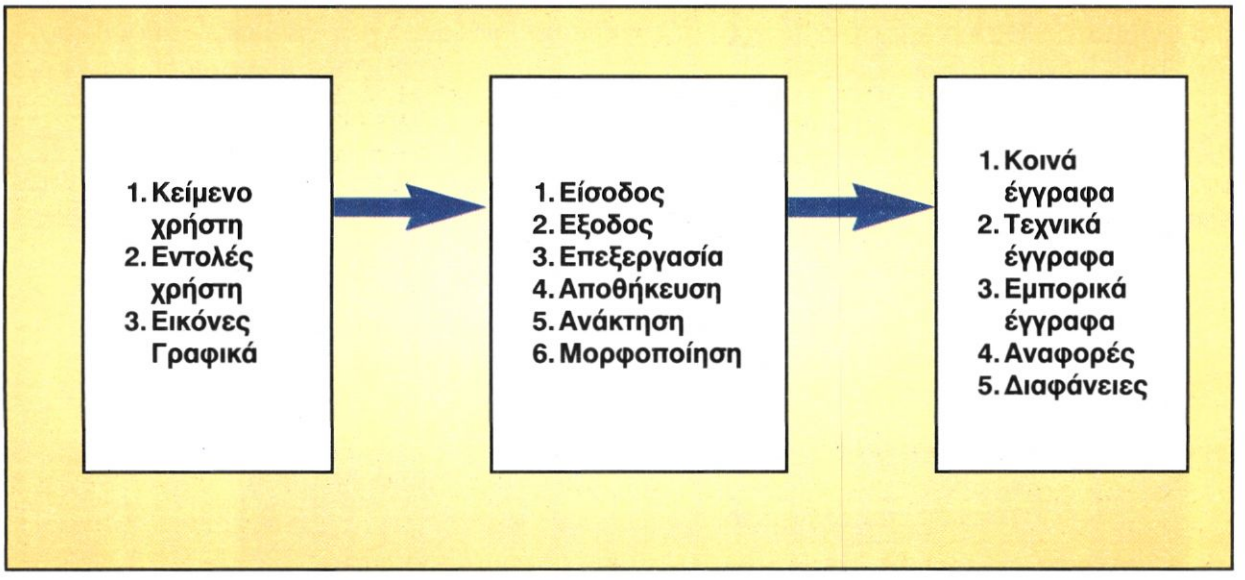
Τα διαγράμματα Ιεραρχίας και Εισόδου - Διεργασίας - Εξόδου *(Hierarchy plus Input - Process - Output, HIPO)* είναι μια διαγραμματική τεχνική που χρησιμοποιείται για την απεικόνιση της εισόδου - εξόδου - λειτουργιών ενός συστήματος ή ενός προγράμματος. Έτσι θεωρείται ως διαγραμματική τεχνική τόσο σε επίπεδο γενικού όσο και λεπτομερειακού σχεδιασμού.
Η σχεδίαση του λογισμικού με διαγράμματα αυτού του τύπου απαιτεί την δημιουργία ενός συνόλου διαγραμμάτων. Αυτά είναι:

α) Ένα **ιεραρχικό διάγραμμα (Η)** που έχει σχήμα δένδρου με κόμβους ορθογώνια παραλληλόγραμμα που συμβολίζουν τις επεξεργασίες του συστήματος (**σχήμα 7.8.**):



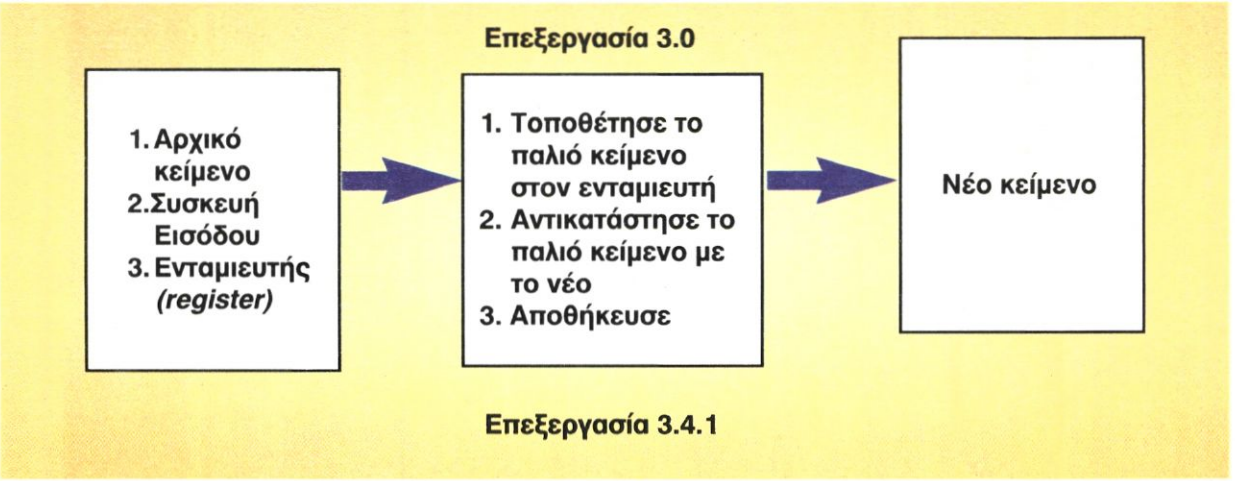
Σχήμα 7.8. - Διάγραμμα Η.

β) Ένα **διάγραμμα γενικής άποψης (IPO)** το οποίο παρέχει καταλόγους μεταβλητών εισόδου, λειτουργιών και μεταβλητών εξόδου (σχήμα 7.9):



Σχήμα 7.9. - Γενικό Διάγραμμα IPO.

γ) Ένα **σύνολο από λεπτομερή διαγράμματα (IPO)**, έναν για κάθε κόμβο του αρχικού ιεραρχικού διαγράμματος που δίνουν λεπτομέρειες για την λειτουργία της αντίστοιχης μονάδας (**σχήμα 7.10**):



Σχήμα 7.10. - Λεπτομερές Διάγραμμα ΙΡΟ.

Η μέθοδος που ακολουθείται είναι η ανάλυση μιας λειτουργίας σε υπολειτουργίες όπως η διεύθυνση ενός οργανισμού σε υποδιευθύνσεις, τμήματα κ.λ.π.

7.2.4. Ψευδοκώδικας

Ο Ψευδοκώδικας χρησιμοποιείται για την ανάλυτική περιγραφή των Μονάδων του Λογισμικού

Σε όλα τα ιεραρχικά διαγράμματα κάθε δομική μονάδα παρουσιάζει μια λειτουργία (ή διαδικασία επεξεργασίας) του υπό μελέτη συστήματος, που, όμως, δεν περιγράφεται στο πλαίσιο του διαγράμματος. Θα λέγαμε ότι οι μονάδες αυτές παρουσιάζονται με τη μορφή "μαύρου κουτιού". Τα περιεχόμενα των μονάδων περιγράφονται αναλυτικά μέσω του **ψευδοκώδικα**. Για το λόγο αυτό ο ψευδοκώδικας αποτελεί μέρος του λεπτομερούς σχεδίου.

Ο ψευδοκώδικας αποτελείται από φράσεις (συνήθως στην προστακτική), γραμμένες στην ελληνική γλώσσα. Με τις φράσεις αυτές περιγράφουμε τις διάφορες λειτουργίες της μονάδας.

Παράδειγμα: Έτσι, για παράδειγμα η μονάδα "εκτύπωση βαθμολογίας μαθητών" περιγράφεται ως εξής:

Μονάδα 3: Εκτύπωση Βαθμολογίας Μαθητών

"Διάβασε τις απαιτήσεις του χρήστη"	(3.1)
"Διάβασε και επεξεργάσου τα δεδομένα του αρχείου"	(3.2)
"Τύπωσε αποτελέσματα"	(3.3)

Αρχικά οι φράσεις είναι γενικές, περιγράφουν σε αρχικό (υψηλό) επίπεδο τις λειτουργίες της μονάδας του λογισμικού και συνοδεύονται από έναν αναγνωριστικό αριθμό όπως και οι μονάδες. Οι παραπάνω φράσεις έχουν αναγνωριστικά 3.1 - 3.3 γιατί είναι η πρώτη , δεύτερη και τρίτη υπολειτουργία της μονάδας 3.

Στη συνέχεια αναλύουμε ακόμα περισσότερο την κάθε φράση σε επόμενο επίπεδο αναλύο-

156

ντας διαρκώς περισσότερο τις λειτουργίες και τις υπολειτουργίες. Έτσι περνάμε από το γενικό στο ειδικό. Για κάθε νέο επίπεδο ανάλυσης προσθέτουμε έναν νέο αριθμό στα αναγνωριστικά των φράσεων, για παράδειγμα:

"Διάβασε και επεξεργάσου τα δεδομένα του αρχείου" ***(3.2)***

While δεν τέλειωσαν τα δεδομένα **(3.2.1)**
    Διάβασε μια εγγραφή από το αρχείο
    Επεξεργάσου την εγγραφή **(3.2.2)**
Endwhile

Επειδή ο ψευδοκώδικας χρησιμοποιείται για τον λεπτομερή σχεδιασμό του λογισμικού γι' αυτό εξαρτάται από τον τρόπο και τα εργαλεία που χρησιμοποιούμε για την υλοποίηση του συστήματος. Έτσι η μορφή του ψευδοκώδικα μπορεί να διαφέρει από εφαρμογή σε εφαρμογή ανάλογα με την γλώσσα προγραμματισμού που χρησιμοποιούμε. Όμως πάντα περιλαμβάνει βασικούς τύπους δομών ελέγχου όπως είναι οι εξής:

Ποιοί είναι οι βασικοί τύποι δομών ελέγχου στον ψευδοκώδικα

• **Επιλογή**

```
IF <ΣΥΝΘΗΚΗ> THEN
  <ΕΝΕΡΓΕΙΕΣ 1>
ELSE
  <ΕΝΕΡΓΕΙΕΣ 2>
```

Εάν η συνθήκη ισχύει (είναι αληθής) τότε εκτέλεσε τις ενέργειες 1 διαφορετικά εκτέλεσε τις ενέργειες 2.

• **Επανάληψη**

```
WHILE <ΣΥΝΘΗΚΗ> DO
ΕΝΕΡΓΕΙΕΣ
ENDWHILE
```

Για όσο ισχύει η συνθήκη εκτέλεσε τις ενέργειες.

```
REPEAT
<ΕΝΕΡΓΕΙΕΣ>
UNTIL <ΣΥΝΘΗΚΗ>
```

Εκτέλεσε τις ενέργειες και για όσο δεν ισχύει η συνθήκη επανέλαβε τις ενέργειες.

```
FOR ΤΙΜΗ_1 ΤΟ ΤΙΜΗ_2 DO
<ΕΝΕΡΓΕΙΕΣ>
```

Εκτέλεσε τις ενέργειες τόσες φορές όσες είναι οι διακεκριμένες τιμές ανάμεσα στην ΤΙΜΗ_1 και την ΤΙΜΗ_Ν (συμπεριλαμβανομένων των τιμών 1 και Ν).

7.3. ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΛΥΣΗ "ΑΠΟ ΠΑΝΩ ΠΡΟΣ ΤΑ ΚΑΤΩ"

Η μεθοδολογία της "Από πάνω προς τα κάτω" σχεδίασης μας βοηθά να αποσυνθέσουμε ένα σχέδιο σε μικρότερες, πιο εύκολα διαχειρίσιμες μονάδες.

Η σχεδίαση είναι η διαδικασία κατά την οποία, με βάση τα δεδομένα της ανάλυσης που έχει προηγηθεί, σχεδιάζεται η λύση του προβλήματος (δηλ. η εφαρμογή λογισμικού). Η σχεδίαση είναι μια νοητική διεργασία που απαιτεί γνώσεις, εμπειρία και φαντασία και γι' αυτό πολλοί την χαρακτηρίζουν και σαν τέχνη. Δεν υπάρχει συγκεκριμένος τρόπος για το πως σχεδιάζουμε το λογισμικό. Γι' αυτό και οι τεχνικές σχεδίασης που έχουν αναπτυχθεί παρέχουν μόνο κατευθυντήριες γραμμές.

Θεμελιώδες χαρακτηριστικό του λογισμικού είναι η δομή του (*structure*). Βασική διαδικασία κατά τη φάση της σχεδίασης είναι η σχεδίαση της δομής του λογισμικού. Η συνήθης τεχνική που χρησιμοποιούμε κατά τη σχεδίαση της δομής είναι η αποσύνθεση ενός μεγάλου συστήματος σε μικρότερες, ευκολότερα διαχειρίσιμες και καλά συσχετισμένες μεταξύ τους μονάδες, οι οποίες θα αντιστοιχούν σε συγκεκριμένες εντολές μιας γλώσσας προγραμματισμού.

ΠΑΡΑΤΗΡΗΣΗ

Η ιδέα της "αποσύνθεσης" δεν εφαρμόζεται μόνο στη σχεδίαση λογισμικού, αλλά σε πολλές άλλες δραστηριότητες, ακόμη και σε δραστηριότητες της καθημερινής ζωής, όπως είναι το παράδειγμα που ακολουθεί.

Παράδειγμα:

Έστω ότι θέλουμε ταξιδέψουμε οδικώς σε μια Ευρωπαϊκή χώρα και να περάσουμε τις διακοπές μας σε συγκεκριμένη λουτρόπολη. Κατ' αρχήν χρησιμοποιούμε ένα χάρτη της Ευρώπης προκειμένου να βρούμε τη διαδρομή που πρέπει να ακολουθήσουμε μέχρι να φτάσουμε στη χώρα που επιλέξαμε. Στη συνέχεια χρησιμοποιούμε ένα χάρτη της χώρας, προκειμένου να βρούμε τη διαδρομή προς τη συγκεκριμένη λουτρόπολη. Τέλος, μπορούμε να χρησιμοποιήσουμε ένα περισσότερο αναλυτικό χάρτη της λουτρόπολης, ο οποίος θα μας δώσει όλες τις χρήσιμες για την εκεί διαμονή μας πληροφορίες. Η διαδικασία αυτή χαρακτηρίζεται "καθοδική" γιατί ξεκινάει από το γενικό και καταλήγει στο ειδικό.

Ανάλογη είναι και η μέθοδος που ακολουθούμε προκειμένου να σχεδιάσουμε το λογισμικό. Η σχεδίαση "**από πάνω προς τα κάτω**" *(top-down design)* μας βοηθά να αποσυνθέσουμε ένα σχέδιο σε μονάδες. Αρχικά σχεδιάζουμε το σύστημα κάνοντας μια μακροσκοπική περιγραφή των δεδομένων εισόδου, της επεξεργασίας που γίνεται από το σύστημα και των αποτελεσμάτων του συστήματος. Η περιγραφή αυτή χαρακτηρίζεται σαν περιγραφή υψηλού επιπέδου. Το προϊόν της περιγραφής αυτής ονομάζεται **σχέδιο επιπέδου 1**. Στη συνέχεια ορίζουμε κάθε μονάδα σαν ένα σύνολο από άλλες, χαμηλότερου επιπέδου μονάδες, δηλαδή μονάδες που εμφανίζουν περισσότερες λεπτομέρειες. Αυτό επαναλαμβάνεται πολλές φορές. Με τον τρόπο αυτό δημιουργούμε το **επίπεδο 2, το επίπεδο 3**, κ.λ.π. Το πλήθος των επιπέδων που θα δημιουργήσουμε εξαρτάται από το μέγεθος και τις επεξεργασίες του συστήματος. Τα επίπεδα αυτά ονομάζονται **επίπεδα αφαίρεσης** (*levels of abstraction*). Με τον προαναφερόμενο τρόπο "τμηματοποιούμε" το

σχέδιο του συστήματος σε βασικές επεξεργασίες με καλά ορισμένη είσοδο και έξοδο. Η δυνατότητα της απεικόνισης του σχεδίου μιας εφαρμογής λογισμικού κατά τμήματα, που κάθε ένα έχει διαφορετικό επιπέδου αφαίρεσης ονομάζεται "**τμηματικότητα**" (*modularity*).

ΠΑΡΑΤΗΡΗΣΗ

Η "τμηματικότητα" μας επιτρέπει να αποκρύψουμε τις τεχνικές λεπτομέρειες του συστήματος στο αρχικό γενικό σχέδιο της εφαρμογής και να επικεντρώσουμε την προσοχή μας στις λειτουργίες της, στο τι ακριβώς κάνει, ποιες είναι οι ροές δεδομένων κ.λ.π.

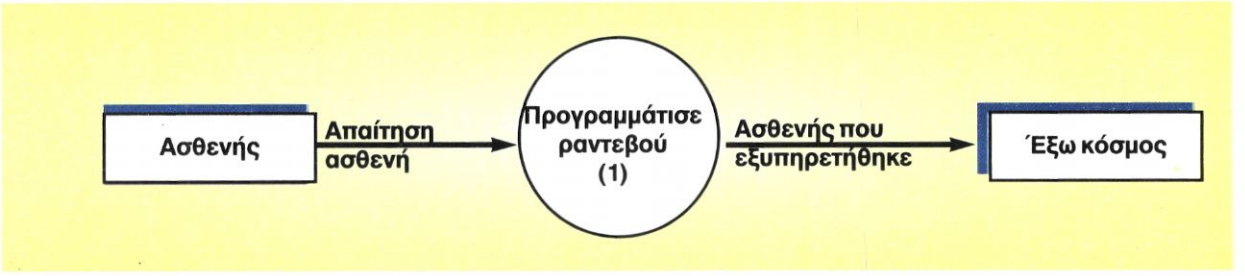
7.3.1. Συναρτησιακή Αποσύνθεση

ορισμός

Όταν η αποσύνθεση αφορά τις επεξεργασίες - λειτουργίες του συστήματος τότε ονομάζεται **συναρτησιακή αποσύνθεση** (*functional decomposition*). Είναι μια μέθοδος που είναι **προσανατολισμένη στη διεργασία** (*process oriented*).

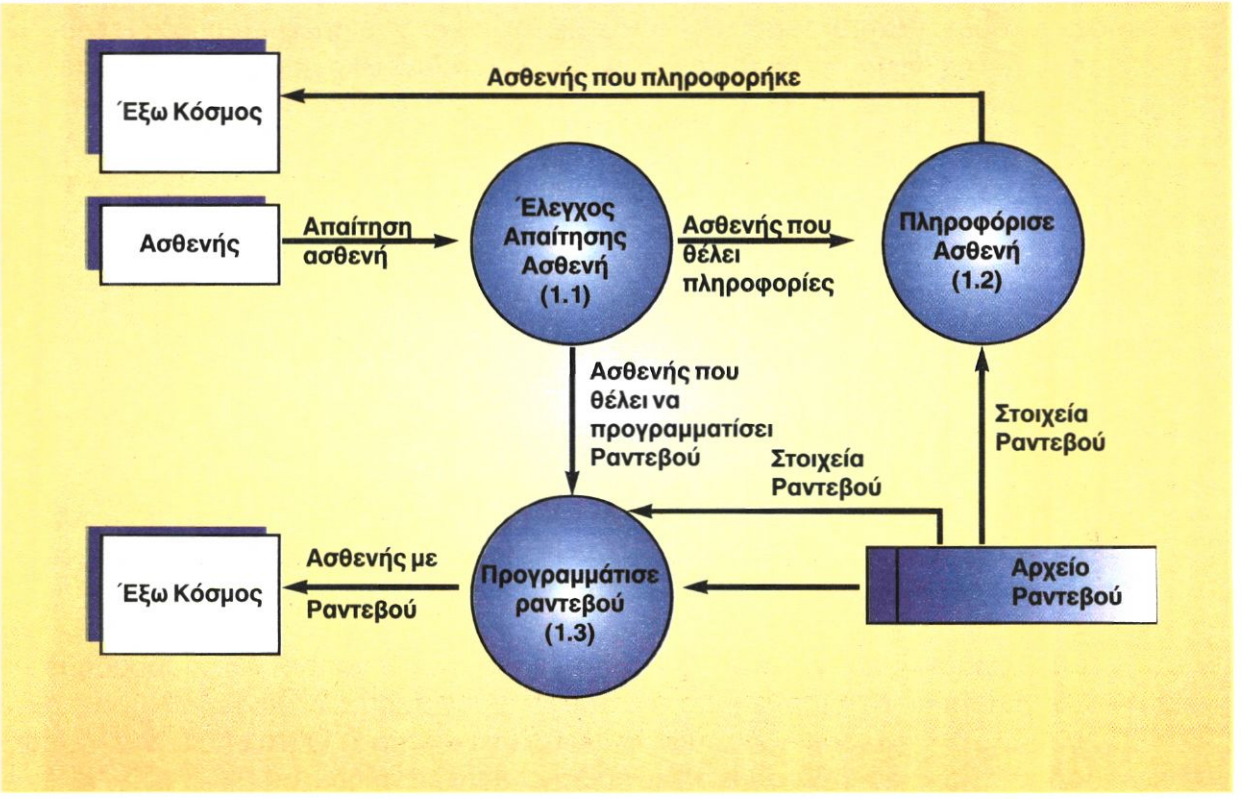
παράδειγμα:

Στο παράδειγμα του "Ιατρείου", που αναφέρθηκε παραπάνω, θα μπορούσαμε να σχεδιάσουμε το υποσύστημα του προγραμματισμού των ραντεβού με ένα διάγραμμα ροής δεδομένων, όπως φαίνεται στο **σχήμα 7.11**, το οποίο, για λόγους απλότητας, ας υποθέσουμε ότι είναι **διάγραμμα επιπέδου 1**.



Σχήμα 7.11. - Διάγραμμα Ροής Δεδομένων Επιπέδου 1.

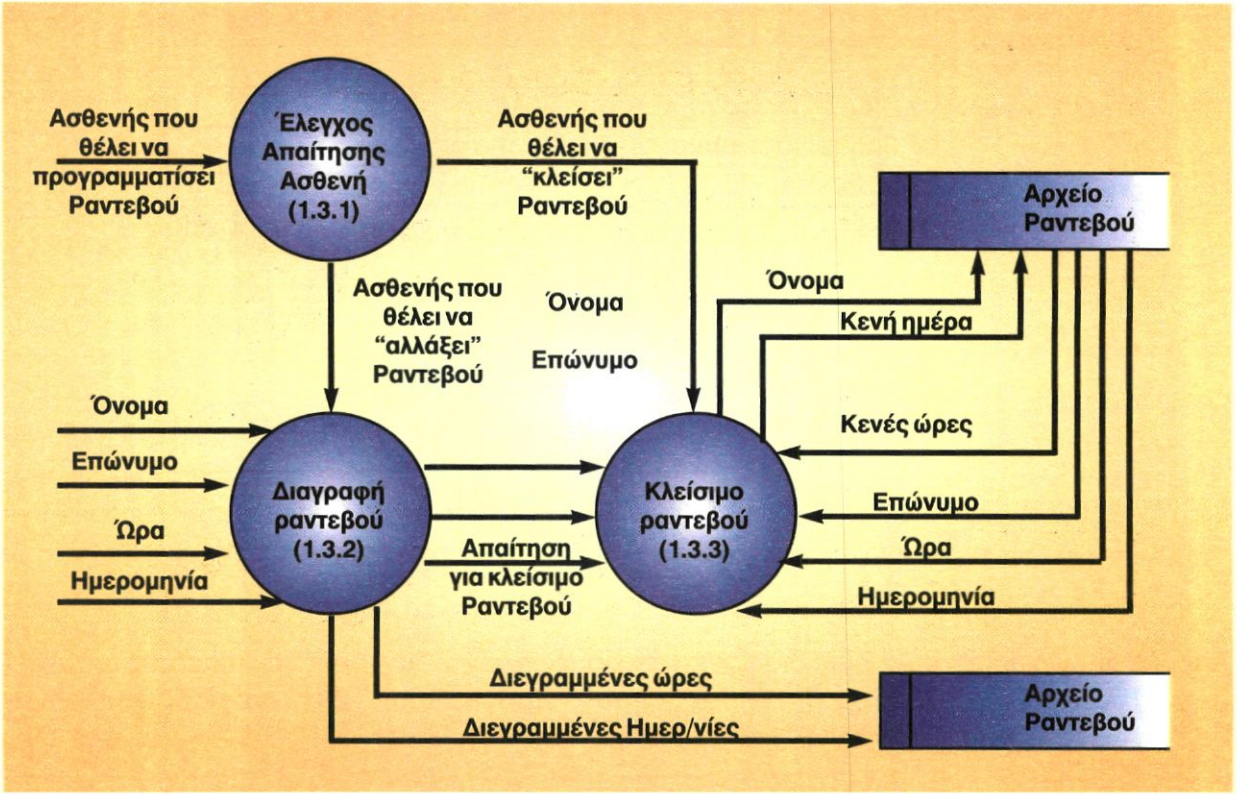
Στη συνέχεια θα ορίσουμε τη μονάδα (1), που απεικονίζεται στο διάγραμμα του παραπάνω σχήματος 7.11., σαν ένα σύνολο από άλλες, χαμηλότερου επιπέδου μονάδες: "Έλεγξε Απαίτηση Ασθενή", "Προγραμμάτισε Ραντεβού" και "Πληροφόρησε Ασθενή" (**σχήμα 7.12**).



Σχήμα 7.12. - Διάγραμμα Ροής Δεδομένων Επιπέδου 2.

Όπως φαίνεται στο παράδειγμα, η επεξεργασία (1) αναλύθηκε στις επεξεργασίες (1.1), (1.2) και (1.3). Με την τεχνική αυτή μπορούμε να επιτύχουμε την τμηματική υλοποίηση από επιμέρους ομάδες ροών δεδομένων, σε κάθε μια από τις οποίες θα απεικονίζεται αναλυτικά ποια είναι η επεξεργασία που θα υλοποιούν, ποια είναι τα δεδομένα εισόδου και ποια τα αποτελέσματα.

Με τον ίδιο τρόπο η επεξεργασία 1.3 ("προγραμμάτισε ραντεβού") μπορεί να αναλυθεί σε ένα διάγραμμα επιπέδου 3 όπως φαίνεται στο **σχήμα 7.13.**



Σχήμα 7.13. - Διάγραμμα Ροής Δεδομένων Επιπέδου 3

Για την αναλυτική περιγραφή των υπομονάδων χρησιμοποιούμε **ψευδοκώδικα**, ο ποίος απεικονίζεται στον **πίνακα 7.2** :

1.3.1	ΕΛΕΓΧΟΣ ΑΠΑΙΤΗΣΗΣ ΑΣΘΕΝΗ
	1.3.1.1 Διάβασε απαίτηση ασθενή
	1.3.1.2 IF η απαίτηση είναι διαγραφή ραντεβού THEN 1.3.1.1.1 CALL διαγραφή ραντεβού ELSE 1.3.1.2 CALL κλείσιμο ραντεβού

1.3.2	ΔΙΑΓΡΑΦΗ ΡΑΝΤΕΒΟΥ
	1.3.2.1 Πάρε απαίτηση ασθενή
	1.3.2.2 Πάρε όνομα και επώνυμο
	1.3.2.3 Πάρε ημερομηνία και ώρα ραντεβού
	1.3.2.4 Κάνε διαγραφή εγγραφής ραντεβού στο αρχείο "ραντεβού"
	1.3.2.5 IF η απαίτηση είναι αλλαγή ραντεβού THEN
	1.3.2.6 1.3.2.6.1 CALL κλείσιμο ραντεβού

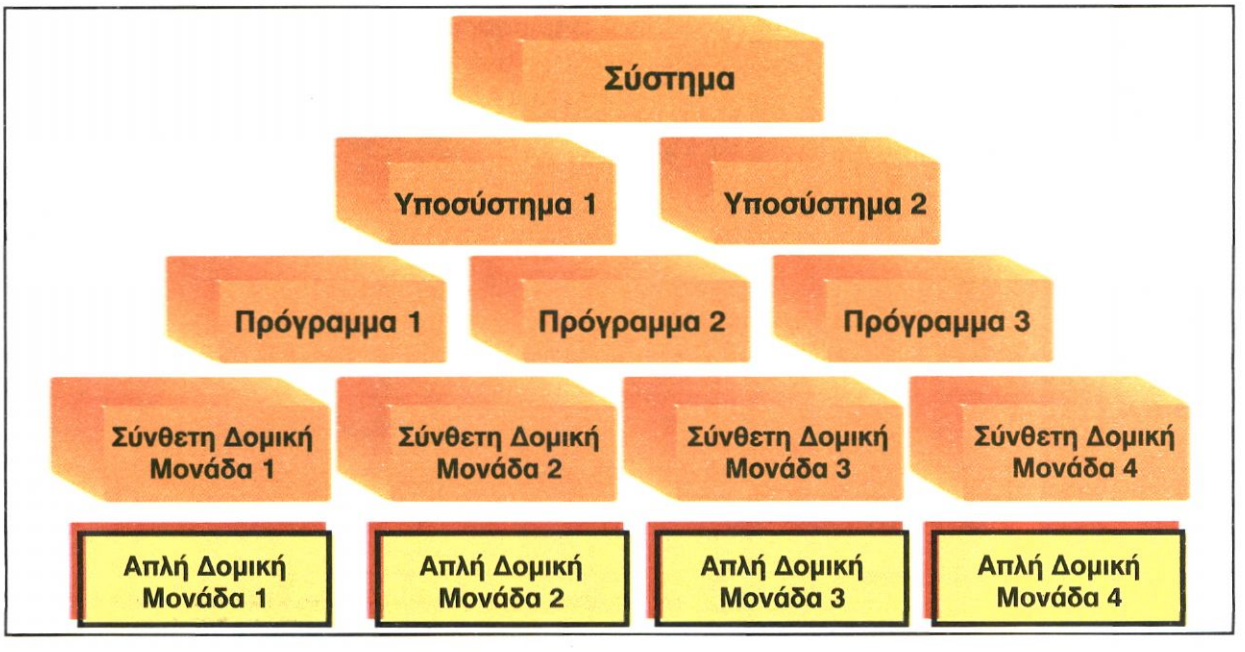
1.3.3	ΚΛΕΙΣΙΜΟ ΡΑΝΤΕΒΟΥ
	1.3.3.1 Πάρε απαίτηση ασθενή
	1.3.3.2 Πάρε όνομα και επώνυμο
	1.3.3.3 Πάρε ημερομηνία και ώρα ραντεβού
	1.3.3.4 Πάρε κενές ώρες ραντεβού από το αρχείο "ραντεβού"
	1.3.3.5 Κάνε κλείσιμο ραντεβού και ενημέρωσε το αρχείο "ραντεβού"

Πίνακας 7.2. - Περιγραφή Υπομονάδων Λογισμικού με Ψευδοκώδικα.

ΠΑΡΑΤΗΡΗΣΗ

Ο ψευδοκώδικας αποτελεί την πιο λεπτομερειακή αποτύπωση του συστήματος, των λειτουργιών του και των μονάδων λογισμικού. Είναι το τελευταίο και το πιο αναλυτικό επίπεδο της ανάλυσης. Αμέσως μετά ακολουθεί η φάση της υλοποίησης όπου οι προγραμματιστές θα μετατρέψουν τον ψευδοκώδικα σε εντολές μιας γλώσσας προγραμματισμού ή γενικότερα ενός εργαλείου ανάπτυξης λογισμικού

Από μια άποψη υπάρχουν πέντε διαφορετικές δομές ή πέντε διαφορετικά επίπεδα αποσύνθεσης του λογισμικού: "**Σύστημα**", "**Υποσύστημα**", "**Πρόγραμμα**", "**Σύνθετες Δομικές Μονάδες**", "**Απλές Δομικές Μονάδες**" (**σχήμα 7.14**):



Σχήμα 7.14. - Δομικές Μονάδες Λογισμικού.

Η τοποθέτηση των κομματιών του λογισμικού σε ιεραρχικά επίπεδα ονομάζεται δόμηση *(structuring)* του λογισμικού. Σχέσεις υπάρχουν μεταξύ των δομικών μονάδων, που βρίσκονται τόσο στο ίδιο όσο και σε διαφορετικά επίπεδα. Ο προσδιορισμός των υποσυστημάτων και των προγραμμάτων από τα οποία αποτελείται ένα σύστημα δεν είναι ούτε ιδιαίτερα δύσκολος, ούτε και κρίσιμος. Το αντίθετο συμβαίνει με τον προσδιορισμό των σύνθετων δομικών μονάδων. Η δυσκολία οφείλεται στο μεγάλο πλήθος των σύνθετων δομικών μονάδων από τις οποίες αποτελείται ένα πρόγραμμα και στο μεγάλο αριθμό διαφορετικών τρόπων που μπορεί να χωριστεί ένα πρόγραμμα σε δομικές μονάδες.

ΠΑΡΑΤΗΡΗΣΗ

Η σχεδίαση σε επίπεδο συστήματος, υποσυστημάτων και προγραμμάτων είναι η **αρχιτεκτονική σχεδίαση** ενώ η σχεδίαση των σύνθετων μονάδων η **λεπτομερειακή σχεδίαση.**

7.3.2. Δομημένη Σχεδίαση

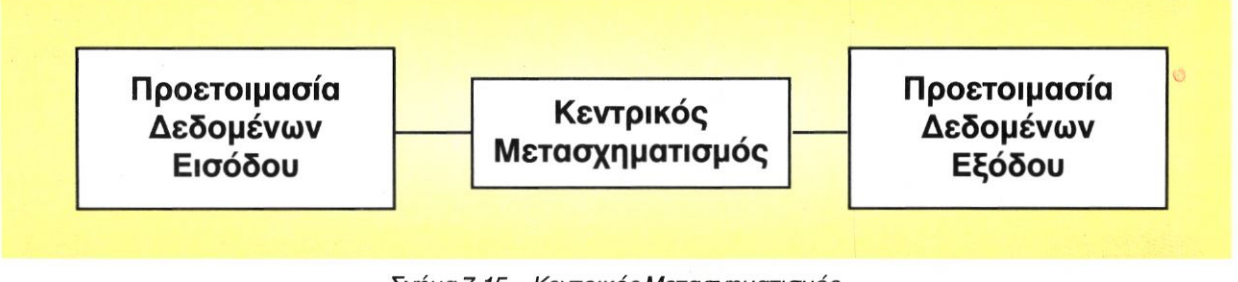
Η **δομημένη σχεδίαση** *(structured design)* είναι μια δημοφιλής τεχνική που ανήκει στην κατηγορία της "από πάνω προς τα κάτω" σχεδίασης. Παρέχει ένα συστηματικό τρόπο για την παραγωγή σχεδίων λογισμικού από τα διαγράμματα ροής δεδομένων.

Ορισμός

Σε αντίθεση με την συναρτησιακή αποσύνθεση, είναι μια μέθοδος **προσανατολισμένη στα δεδομένα** *(data oriented)* και εξετάζει τη ροή των δεδομένων. Η αποτύπωση των σχεδίων λογισμικού γίνεται με τα διαγράμματα δομής.

Βασικές έννοιες σε αυτή τη μεθοδολογία είναι οι εξής:

- **Κεντρικός μετασχηματισμός**: ένα σύνολο επεξεργασιών που εκτελεί επεξεργασία δεδομένων εισόδου και παράγει δεδομένα εξόδου. Τα δεδομένα εισόδου προετοιμάζονται από ένα σύνολο επεξεργασιών που προηγούνται του κεντρικού μετασχηματισμού. Τα δεδομένα εξόδου προετοιμάζονται από ένα σύνολο επεξεργασιών του έπονται του κεντρικού μετασχηματισμού (**σχήμα 7.15**).



Σχήμα 7.15. - Κεντρικός Μετασχηματισμός.

• **Κέντρο δοσοληψιών** (*transaction center*): σ' ένα διάγραμμα ροής δεδομένων μια επεξεργασία χαρακτηρίζεται κέντρο δοσοληψιών όταν δέχεται δεδομένα εισόδου από ένα σύνολο επεξεργασιών που προηγούνται και κατανέμει τη ροή σε πολλά κέντρα επεξεργασίας (**σχήμα 7.16**).

Σχήμα 7.16. - Κέντρο Δοσοληψιών.

Ένα τμήμα του διαγράμματος ροής δεδομένων που περιέχει κεντρικό μετασχηματισμό ή κέντρο δοσοληψιών ονομάζεται **χαρακτηριστικό τμήμα**.

Το λογισμικό που σχεδιάζουμε πρέπει να διαθέτει υψηλή ανεξαρτησία των σύνθετων δομικών των μονάδων δηλαδή χαλαρή σύζευξη μεταξύ των μονάδων και υψηλή συνθετικότητα κάθε μονάδας.

• **Ανεξαρτησία σύνθετων δομικών μονάδων:** η έννοια της ανεξαρτησίας σχετίζεται με τις έννοιες της **σύζευξης** (*compling*) και της **συνεκτικότητας** (*cohesion*).
Η σύζευξη μεταξύ των δομικών μονάδων δημιουργείται για κάποιον από τους παρακάτω λόγους:

α) Πέρασμα διαμέσου παραμέτρων από την μια μονάδα στην άλλη **δεδομένων υπολογισμού**. Η σύζευξη αυτή προκαλεί **χαμηλή εξάρτηση** ανάμεσα στις δομικές μονάδες.

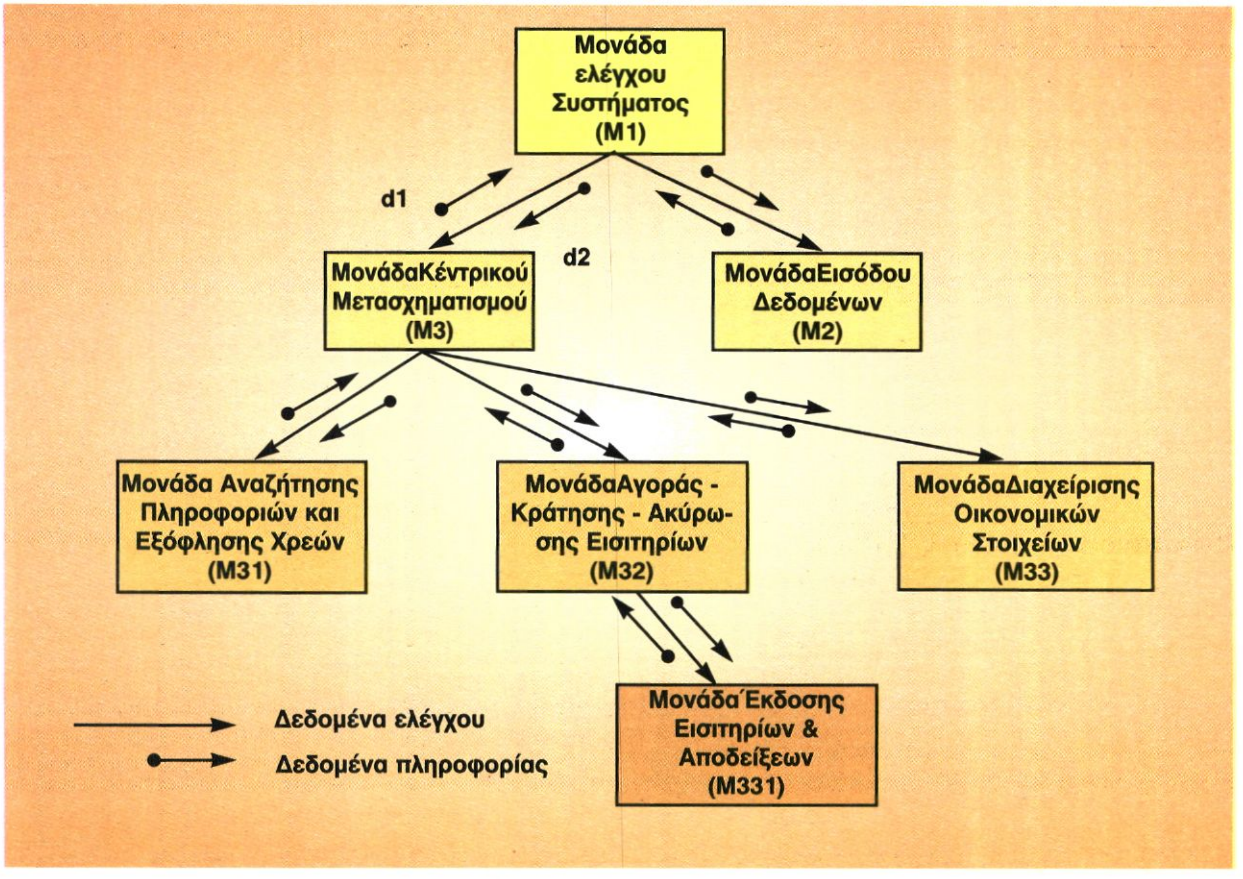
β) Πέρασμα δια μέσου παραμέτρων από την μια μονάδα στην άλλη **δεδομένων ελέγχου** που καθορίζουν τις λειτουργίες που εκτελούνται και την ροή του προγράμματος. Η σύζευξη αυτή προκαλεί **υψηλή εξάρτηση** ανάμεσα στις δομικές μονάδες.

γ) Χρησιμοποίηση και από τις δυο μονάδες **κοινού πόρου δεδομένων** όπως αρχείου, δομής δεδομένων κ.λ.π. Η σύζευξη αυτή προκαλεί **πολύ υψηλή εξάρτηση** ανάμεσα στις δομικές μονάδες.

Η συνεκτικότητα αφορά στο πλήθος των λειτουργιών (επεξεργασιών) που εκτελεί μαι δομική μονάδα. Γενικά ισχύει ότι όσο μικρότερο είναι το πλήθος των λειτουργιών που εκτελεί η μονάδα τόσο μεγαλύτερη είναι η συνεκτικότητα της μονάδας.

ΠΑΡΑΤΗΡΗΣΗ

Η τεχνική της δομημένης σχεδίασης περιλαμβάνει **τέσσερα βήματα**. Στόχος μας κατά την διαδικασία αυτή είναι να δημιουργήσουμε τα κατάλληλα **διαγράμματα δομής λογισμικού** χρησιμοποιώντας τα διαγράμματα ροής δεδομένων που δημιουργήσαμε κατά τη φάση της ανάλυσης απαιτήσεων.



Σχήμα 7.18. - Διάγραμμα Δομής Συστήματος (Αρχιτεκτονικό Σχέδιο).

Βήμα Τέταρτο: Στο βήμα αυτό επιχειρείται ο **διαμελισμός** (και μερικές φορές η **συνένωση**) των **λειτουργιών** (κουτιών) του διαγράμματος δομής έτσι ώστε να παραχθεί μια καλύτερη προσέγγιση του αρχιτεκτονικού σχεδίου. Τα κριτήρια που χρησιμοποιούνται εδώ είναι η **σύζευξη** και η **συνεκτικότητα**. Το βήμα αυτό επαναλαμβάνεται όσες φορές κρίνεται σκόπιμο ότι είναι αναγκαίο να γίνει.

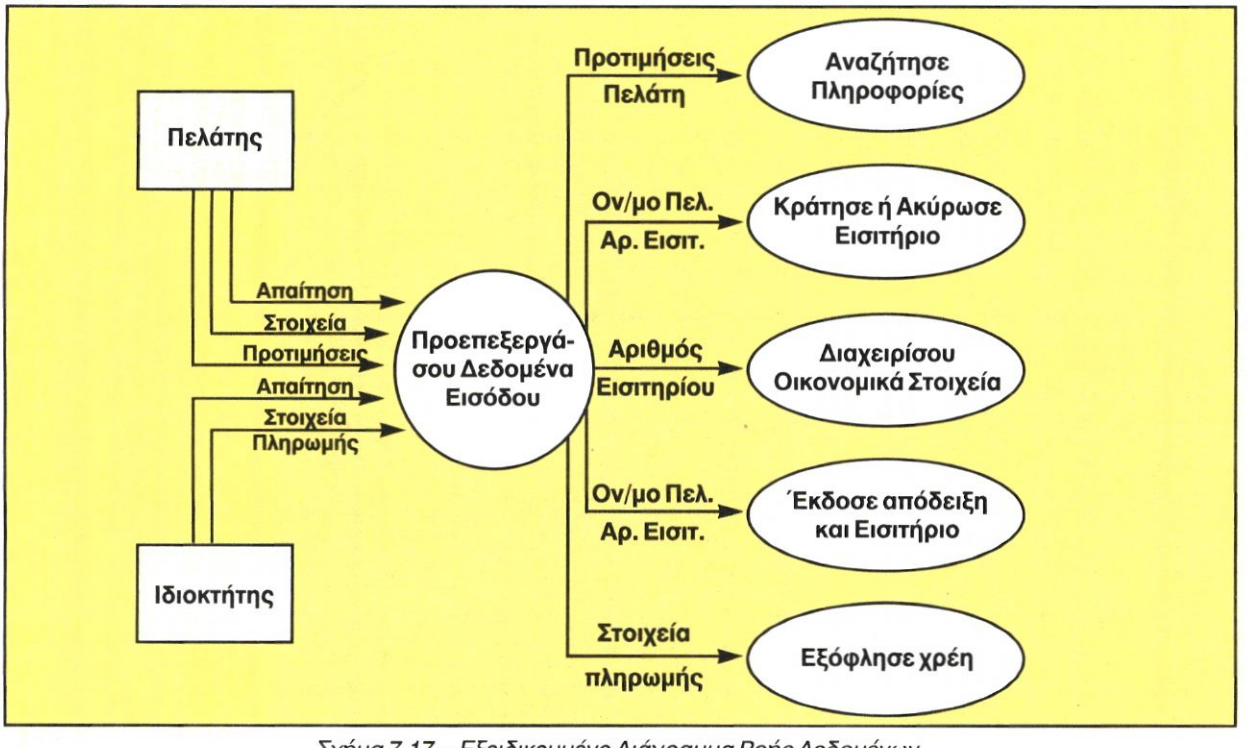
Στην συνέχεια πρέπει να γίνει **γενική περιγραφή των μονάδων του αρχιτεκτονικού σχεδίου**. Η περιγραφή πρέπει να περιλαμβάνει για κάθε μονάδα τα εξής χαρακτηριστικά: **Ταυτότητα, Τύπο, Σκοπό, Λειτουργία, Υφιστάμενες Μονάδες.**

Για παράδειγμα, τα χαρακτηριστικά της μονάδας M1 του σχήματος 7.14 περιγράφονται στον παρακάτω **πίνακα 7.3.**

Βήμα Πρώτο: Αρχικά συμπληρώνονται και εξειδικεύονται τα διαγράμματα ροής δεδομένων που έχουν παραχθεί κατά την φάση της ανάλυσης των απαιτήσεων. Η εξειδίκευση αφορά την **διεπαφή** (*interface*) ανάμεσα στο σύστημα και τον χρήστη και τον έλεγχο της ορθότητας των δεδομένων και των ροών που απεικονίζονται.

Έτσι για παράδειγμα στο βήμα αυτό παράγεται το ακόλουθο διάγραμμα ροής δεδομένων (**σχήμα 7.17**) για το σύστημα "Γραφείο έκδοσης αεροπορικών εισιτηρίων" (**σχήμα 7.2**):

Παράδειγμα:



Σχήμα 7.17. - Εξειδικευμένο Διάγραμμα Ροής Δεδομένων.

Βήμα Δεύτερο: Στο βήμα αυτό **εντοπίζονται τα χαρακτηριστικά τμήματα των διαγραμμάτων ροής δεδομένων**. Έτσι κάθε διάγραμμα χωρίζεται σε χαρακτηριστικά τμήματα. Το παραπάνω διάγραμμα ροής δεδομένων (**σχήμα 7.17**) είναι ένα χαρακτηριστικό τμήμα διαγράμματος ροής δεδομένων γιατί παρουσιάζει έναν κεντρικό μετασχηματισμό.

Βήμα Τρίτο: Στο βήμα αυτό **κάθε χαρακτηριστικό τμήμα του διαγράμματος ροής δεδομένων απεικονίζεται σ' ένα τμήμα του διαγράμματος δομής** του συστήματος. Κάθε μετασχηματισμός στο διάγραμμα ροής απεικονίζεται σε μια σύνθετη δομική μονάδα. Έτσι παίρνουμε μια πρώτη προσέγγιση του αρχιτεκτονικού σχεδίου του συστήματος (**σχήμα 7.18**).

Δεν θα πρέπει να συγχέεται ο έλεγχος του σχεδίου του συστήματος με τον έλεγχο σφαλμάτων του κώδικα και τους λοιπούς τελικούς ελέγχους του λογισμικού, που θα αναφερθούν στα επόμενα κεφάλαια

Περιγραφή Μονάδας Μ1	
Ταυτότητα	Μονάδα Ελέγχου Συστήματος
Τύπος	Μονάδα Προγράμματος (Κύριο Πρόγραμμα)
Σκοπός	Βασικός σκοπός της μονάδας Μ1 είναι η μεταφορά των απαραίτητων δεδομένων από την μονάδα Μ2 στην μονάδα Μ3. Επίσης η Μονάδα Μ1 χειρίζεται και όλους τους καταλόγους επιλογών *(Menu)* που είναι η διεπαφή με τον χρήστη.
Λειτουργία	Η μονάδα Μ1 αρχικοποιεί τις καθολικές δομές που ορίζονται στο πρόγραμμα, εμφανίζει τον αρχικό κατάλογο επιλογών και ανάλογα με τις επιλογές του χρήστη καλεί τα αντίστοιχα προγράμματα. Επί σης καλεί τη μονάδα Μ2 με ένα δεδομένο ελέγχου *(flag)* και λαμβάνει από αυτήν τα δεδομένα που χρειάζεται. Τέλος, αν χρειαστεί, στέλνει τα δεδομένα που χρειάζονται στην Μ3 για την έναρξη της λειτουργίας της.
Υφιστάμενες Μονάδες	Η μονάδα Μ1 βρίσκεται στην κορυφή (ρίζα) του αρχιτεκτονικού σχεδίου και επομένως όλες οι λοιπές μονάδες είναι υφιστάμενες της Μ1.

Πίνακας 7.3. - Περιγραφή Χαρακτηριστικών Μονάδας Λογισμικού.

Μετά την γενική περιγραφή των μονάδων γίνεται η **αναλυτική περιγραφή των δεδομένων** που εμφανίζονται στο αρχιτεκτονικό σχέδιο. Εδώ πρέπει να καθορίσουμε τον τύπο των δεδομένων και την λειτουργία τους (όπου κρίνεται απαραίτητο).

Για παράδειγμα τα δεδομένα d1 και d2 που εμφανίζονται στο αρχιτεκτονικό σχέδιο.

Περιγραφή Οντότητας Δεδομένων d1		
d1.1	Στοιχεία Πελάτη	Ονοματεπώνυμο, Διεύθυνση, Τηλέφωνο
d2.2	Προτιμήσεις Πελάτη	Όνομα Αεροπορικής Εταιρείας, Προορισμός Πτήσης, Ημερομηνία Πτήσης, Κωδικός Εισιτηρίου
d3.3	Στοιχεία Πληρωμής	Όνομα Αεροπορικής Εταιρείας, Ποσό πληρωμής
Περιγραφή Οντότητας Δεδομένων d2		
D2	Δεδομένο Ελέγχου που παίρνει τιμές από 0 έως 4	0: Αγορά Εισιτηρίου 1: Κράτηση Εισιτηρίου 2: Αναζήτηση Πληροφοριών 3: Εξόφληση Χρέους 4. Ακύρωση Εισιτηρίου

Πίνακας 7.4. - Περιγραφή των Δεδομένων του Αρχιτεκτονικού Σχεδίου.

Μετά την ολοκλήρωση του αρχιτεκτονικού σχεδιασμού ακολουθεί η φάση κατά την οποία δημιουργούμε το **λεπτομερές σχέδιο** του συστήματος που περιλαμβάνει την αναλυτική περιγραφή των σύνθετων μονάδων του λογισμικού όπως αυτές εμφανίζονται στο αρχιτεκτονικό σχέδιο.

Το λεπτομερές σχέδιο επεκτείνει το αρχιτεκτονικό σχέδιο αλλά διαφέρει σημαντικά από αυτό. Περιέχει λεπτομέρειες για τα δεδομένα, τις δομές δεδομένων και τους αλγόριθμους που θα χρησιμοποιήσουμε. Για παράδειγμα το αρχιτεκτονικό σχέδιο μπορεί να προσδιορίζει την χρήση ενός αρχείου από μια μονάδα του. Το λεπτομερές σχέδιο καθορίζει τον τρόπο οργάνωσης του αρχείου καθώς και τον τρόπο προσπέλασής του.

Τύποι δεδομένων - Δεδομένα

Ο καθορισμός των δεδομένων πρέπει να γίνεται με τέτοιο τρόπο, ώστε να παρέχεται ευκολία στην υλοποίηση και ασφαλή λειτουργία του συστήματος, ανεξάρτητα από τους χειρισμούς του χρήστη. Αυτό βρίσκεται σε συνάρτηση με τη συνέπεια και την αξιοπιστία του συστήματος.

Βασικά χαρακτηριστικά των αλγόριθμων είναι η ορθότητα και η αποδοτικότητα:

Ορθότητα αλγορίθμου:

Οι αλγόριθμοι που χρησιμοποιούμε πρέπει να είναι σωστοί και να μας οδηγούν στα αναμενόμενα αποτελέσματα, με βάση τις προϋποθέσεις που έχουμε καθορίσει. Για να είναι ένας αλγόριθμος σωστός πρέπει, κάτω από οποιεσδήποτε συνθήκες (τιμές και τύποι δεδομένων, σειρά εκτέλεσης λειτουργιών κ.λ.π.), να δίνει πάντα τα αναμενόμενα αποτελέσματα.

Αποδοτικότητα Αλγόριθμου:

Ένας αλγόριθμος χρειάζεται πόρους, όπως για παράδειγμα μνήμη, χρόνο επεξεργασίας, κ.λ.π. Ο βαθμός απόδοσης ενός αλγόριθμου είναι αντίστροφα ανάλογος των ποσοτήτων των πόρων που χρειάζεται. Για παράδειγμα, όσο μικρότερος είναι ο συνολικός αριθμός των συγκρίσεων των δεδομένων ή όσο μικρότερο το μέγεθος και το πλήθος των προσωρινών αρχείων ή των δομών δεδομένων που χρησιμοποιεί, τόσο μεγαλύτερη είναι η αποδοτικότητα του.

ΠΑΡΑΤΗΡΗΣΗ

Για την αποτύπωση του λεπτομερούς σχεδίου χρησιμοποιούμε τις ίδιες διαγραμματικές τεχνικές που περιγράψαμε στο κεφάλαιο 7.3, δηλαδή τα διαγράμματα HIPO και ο ψευδοκώδικας.

Στον παρακάτω πίνακα 7.5 δίνεται σαν παράδειγμα ο ψευδοκώδικας για την μονάδα M1:

Λεπτομερές Σχέδιο Μονάδας	M1
Ταυτότητα Μονάδας	Μονάδα Ελέγχου Συστήματος
Πρόγραμμα	1
Ταυτότητα Προγράμματος	Main
Εσωτερικά Δεδομένα	Int Επιλογή1, Επιλογή2, Επιλογή3 Char Flag, Flag1

Όρισε τις κατάλληλες δομές και μεταβλητές

```
Repeat
   Εμφάνισε τον κατάλογο επιλογών (menu)
   Διάβασε την επιλογή του χρήστη (επιλογή1)
   Case επιλογη1 of
      1: Call M3
      2: Εμφάνισε τον κατάλογο επιλογών (menu2)
        Διάβασε την επιλογή του χρήστη (επιλογή2)
         Flag= επιλογή2
         Call M2 (flag)
         Call M3 (flag)
      3: Εμφάνισε τον κατάλογο επιλογών (menu3)
        Διάβασε την επιλογή του χρήστη (επιλογή3)
         Flag1= επιλογή3
         Call M2 (flag1)
         Call M3 (flag1)
Until επιλογή1= 0
```

Πίνακας 7.5. - Ψευδοκώδικας Μονάδας Λογισμικού.

7.3.1. Σύνθεση των Εργασιών σε Τελικό Διάγραμμα

Μετά την ολοκλήρωση του αρχιτεκτονικού και του λεπτομερούς σχεδιασμού γίνεται η σύνθεση των επί μέρους εργασιών σε ένα τελικό διάγραμμα. Κάθε ομάδα σχεδιαστών που συμμετείχε στη φάση του σχεδιασμού θα πρέπει να παραδώσει ένα μέρος του σχεδίου στον υπεύθυνο για το σχεδιασμό του συστήματος. Μετά την παράδοση των επιμέρους τμημάτων γίνεται η σύνθεση σε ένα τελικό διάγραμμα.

Με το τελικό διάγραμμα μπορούμε να ελέγξουμε συνολικά το σύστημα, τις λειτουργίες του, τις ροές των δεδομένων, να εντοπίσουμε τα στοιχεία που πρέπει να τροποποιήσουμε και να προχωρήσουμε στον επανασχεδιασμό οντοτήτων – επεξεργασιών, εφόσον αυτό κριθεί απαραίτητο. Ο έλεγχος περιλαμβάνει την επανεξέταση του αρχιτεκτονικού και του λεπτομερούς σχεδίου και ολοκληρώνεται με την επικύρωση των σχεδίων αυτών. Επίσης, το τελικό διάγραμμα χρησιμοποιείται και σαν προϊόν εισόδου στην επόμενη φάση, που είναι η φάση της υλοποίησης. Με βάση το τελικό διάγραμμα γίνεται η ανάθεση των εργασιών στις επί μέρους ομάδες κωδικοποίησης.

Ανακεφαλαίωση / περίληψη

Το λογισμικό των ηλεκτρονικών υπολογιστών είναι ένα εξαιρετικά πολύπλοκο κατασκεύασμα. Η κατασκευή (ανάπτυξη) ποιοτικού, αποτελεσματικού και απαλλαγμένου λογικών σφαλμάτων λογισμικού προϋποθέτει τον καλό σχεδιασμό του.

Η σχεδίαση περιλαμβάνει τη δημιουργία του αρχιτεκτονικού σχεδίου και του λεπτομερούς σχεδίου. Το αρχιτεκτονικό σχέδιο, που ονομάζεται και προκαταρκτικό, περιλαμβάνει τη δημιουργία του εξωτερικού και του εσωτερικού σχεδίου του υπό μελέτη συστήματος.

Οι διαγραμματικές τεχνικές που χρησιμοποιούνται είναι πολλές. Η επιλογή συγκεκριμένης τεχνικής εξαρτάται από το σύστημα που σχεδιάζουμε και τη μεθοδολογία σχεδίασης που ακολουθούμε. Η αποσύνθεση του σχεδίου του συστήματος ονομάζεται "σχεδίαση και ανάλυση από πάνω προς τα κάτω" και οι μεθοδολογίες που χρησιμοποιούμε είναι η "συναρτησιακή αποσύνθεση" και η "δομημένη σχεδίαση". Ο σχεδιασμός ολοκληρώνεται με τη σύνθεση των εργασιών σε ένα τελικό διάγραμμα και τον έλεγχο του σχεδίου του συστήματος.

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Πότε γίνεται η σχεδίαση του λογισμικού. Ποιες φάσεις προηγούνται και ποιες έπονται;
2. Εξηγείστε τη διαφορά μεταξύ εξωτερικού και εσωτερικού αρχιτεκτονικού σχεδίου και λεπτομερούς σχεδίου.
3. Ποιες είναι οι διαγραμματικές τεχνικές της γενικής περιγραφής του λογισμικού και ποιες της λεπτομερειακής περιγραφής;
4. Ποιες δομημένες διαγραμματικές τεχνικές γνωρίζετε. Εκτός από αυτές που αναφέρονται αναζητείστε από διαφορετικές πηγές (βιβλία στην βιβλιοθήκη, διαδίκτυο) άλλες δομημένες διαγραμματικές τεχνικές περιγραφής και σχεδίασης λογισμικού και παρουσιάστε τις συνοπτικά στην τάξη.
5. Ποια είναι τα βασικά στοιχεία ενός διαγράμματος ροής δεδομένων;
6. Τι είναι ο ψευδοκώδικας και που χρησιμοποιείται;
7. Ποιες είναι οι τεχνικές που χρησιμοποιούμε στην "από πάνω προς τα κάτω σχεδίαση"; Ποιες είναι οι βασικές διαφορές των τεχνικών αυτών;
8. Ποιες είναι οι δομικές μονάδες του λογισμικού;
9. Δώστε - περιγράψτε ένα παράδειγμα χαμηλής εξάρτησης και ένα παράδειγμα υψηλής εξάρτησης ανάμεσα σε δύο μονάδες λογισμικού.
10. Περιγράψτε την τεχνική της δομημένης σχεδίασης. Ποια βήματα περιλαμβάνει;
11. Περιγράψτε τους ακόλουθους όρους, κάνοντας και χρήση σχετικών παραδειγμάτων, τόσο από τον χώρο των εφαρμογών λογισμικού όσο και από άλλους γνωστικούς χώρους.

- Αλγόριθμος
- Διεπαφή
- Ενότητα
- Συναρτησιακή αποσύνθεση
- Σύζευξη
- Συνεκτικότητα
- Επίπεδα αφαίρεσης

Γλωσσάριο - Ευρετήριο Όρων

αρχείο *(file)* - Ένας χώρος αποθήκευσης δεδομένων ή εγγραφών που λαμβάνεται σαν ενιαία και αυτοτελής ομάδα [152].

αφορμή εκκίνησης *(trigger)* - Ένα γεγονός - συμβάν που προκαλεί την άμεση εκτέλεση ενός προγράμματος [151].

διάγραμμα δομής *(structure chart)* - Ένα ιεραρχικό διάγραμμα που παρουσιάζει τη συνολική αρχιτεκτονική δομή ενός προγράμματος [154].

επεξεργασία *(process)* - Τμήμα του συστήματος, το οποίο μετασχηματίζει δεδομένα εισόδου (inputs) σε δεδομένα εξόδου *(outputs)* [151].

κεντρικός μετασχηματισμός - Ένα σύνολο επεξεργασιών που εκτελεί επεξεργασία δεδομένων εισόδου και παράγει δεδομένα εξόδου [163].

ροή δεδομένων *(data flow)* - Κανάλι μέσα από το οποίο περνούν τα δεδομένα σε ένα διάγραμμα ροής δεδομένων [151].

σχεδίαση *(design)* - Διαδικασία κατά την οποία δημιουργούμε ένα ομοίωμα της πραγματικής κατασκευής, το οποίο ονομάζεται σχέδιο [147].

Βιβλιογραφία

Stevens W., "Using Structured Design: How to Make Programs Simple", **Changeable**, Flexible and Reusable, Wiley-Interscience, 1981.

Constantine E. and L., "Structured Design: Fundamentals of A Discipline of Computer Program and Systems Design", Prentice-Hall, 1979.

DeMarco T., "Structured Analysis and System Specification", Yourdon Press, 1978.

Κιουντούζη Ε., "Μεθοδολογίες Ανάλυσης & Σχεδιασμού Συστημάτων", Εκδόσεις Ε. Μπένου, 1997.

ΚΕΦΑΛΑΙΟ 8

ΑΝΑΖΗΤΗΣΗ ΚΑΙ ΕΠΙΛΟΓΗ ΤΩΝ ΚΑΤΑΛΛΗΛΩΝ ΕΡΓΑΛΕΙΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

8.1. Επιλογή των εργαλείων που θα χρησιμοποιηθούν για τη λύση των επιμέρους προβλημάτων

8.2. Επιλογή των εφαρμογών στις οποίες θα ενσωματωθούν οι λύσεις

8.3. Διερεύνηση της δυνατότητας συνεργασίας και ανταλλαγής δεδομένων μεταξύ διαφορετικών εφαρμογών

ΕΙΣΑΓΩΓΗ

Πριν αρχίσει ο αναλυτής να αναπτύσσει ένα σύστημα, πρέπει να επιλέξει ανάμεσα στα εργαλεία που υπάρχουν για το σκοπό αυτό, εκείνα που του παρέχουν τα περισσότερα πλεονεκτήματα, σε σχέση με το πρόβλημα που αντιμετωπίζει.

- Ποια είναι, ομως, τα κριτήρια με τα οποία θα γίνει αυτή η επιλογή;
- Ποια ερωτήματα θα πρέπει προηγουμένως να απαντηθούν;

Επειδή, όπως έχει εξηγηθεί και στα κεφάλαια 4 και 5, ένα πρόβλημα συχνά διαιρείται σε επιμέρους υποπροβλήματα, είναι δυνατό ο αναλυτής να επιλέξει διαφορετικό εργαλείο για κάθε επιμέρους πρόβλημα. Το κριτήριο επιλογής το δίνει κάθε φορά μια σειρά παραμέτρων, που έχουν σχέση με τη λειτουργία και τις απαιτήσεις της εφαρμογής. Οι παράμετροι αυτές περιγράφονται και εξηγούνται αναλυτικά στο παρόν κεφάλαιο.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ - ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να γίνει αντιληπτό από το μαθητή, ότι η επιλογή των εργαλείων ανάπτυξης ενός συστήματος δεν αποτελεί προφανή διαδικασία, αλλά αντίθετα, απαιτεί συνεκτίμηση πολλών παραγόντων, ώστε να επιλεγούν τελικά τα καταλληλότερα εργαλεία για το συγκεκριμένο σύστημα.
2. Να μπορεί ο μαθητής να προσδιορίζει, ανάλογα με το είδος του συστήματος, ποιος τύπος λογισμικού ανάπτυξης είναι προτιμότερο να επιλεγεί, καθώς και τις τυχόν εναλλακτικές επιλογές.
3. Να μπορεί να καταλήγει σε συγκεκριμένα εργαλεία ανάπτυξης, τα οποία να είναι κατάλληλα για τις συνθήκες και το περιβάλλον ανάπτυξης της εφαρμογής, λαμβάνοντας παράλληλα υπόψη τη δυνατότητα ανταλλαγής δεδομένων με άλλες εφαρμογές.

8.1. ΕΠΙΛΟΓΗ ΤΩΝ ΕΡΓΑΛΕΙΩΝ ΠΟΥ ΘΑ ΧΡΗΣΙΜΟΠΟΙΗΘΟΥΝ ΓΙΑ ΤΗ ΛΥΣΗ ΤΩΝ ΕΠΙΜΕΡΟΥΣ ΠΡΟΒΛΗΜΑΤΩΝ

Εφόσον η ανάλυση του συστήματος έχει ολοκληρωθεί και ο σχεδιασμός έχει προχωρήσει στο τελικό στάδιο, δημιουργείται η ανάγκη να εξεταστεί **τι είδους εργαλεία απαιτούνται** προκειμένου το τελικό προϊόν να καλύπτει τις αρκετά σαφείς, πλέον, απαιτήσεις, που θέτει το σύστημα. Για να βοηθηθεί κανείς σε μια σωστή επιλογή, υπάρχουν μερικά απλά ερωτήματα στα οποία πρέπει να απαντήσει:

- Πρόκειται για εφαρμογή, η οποία απαιτεί **ανεξαρτησία των δεδομένων** από τα προγράμματα, που τα διαχειρίζονται; Υπάρχουν, για παράδειγμα, περιπτώσεις, όπου απαιτείται το σύστημα να παρέχει τη δυνατότητα στους χρήστες να αντλούν πληροφορίες με διάφορους τρόπους και συνδυάζοντας διάφορα κριτήρια αναζήτησης, χωρίς να είναι απαραίτητη κάθε φορά η κατασκευή νέου προγράμματος.

- Είναι μείζον ζήτημα το θέμα της **ασφάλειας** και της **ακεραιότητας** των δεδομένων; Με τον όρο "ασφάλεια δεδομένων" εννοούμε τη διασφάλισή τους από ανεπιθύμητη πρόσβαση, είτε κάποιου χρήστη του συστήματος, ο οποίος δεν πρέπει να έχει πρόσβαση σε συγκεκριμένες πληροφορίες, είτε κάποιου τρίτου, ο οποίος για παράδειγμα μέσα από το διαδίκτυο θα καταφέρει να διεισδύσει παράνομα στο σύστημα και να αντλήσει ή να καταστρέψει πληροφορίες. Εξασφάλιση της ακεραιότητας των δεδομένων ενός συστήματος, σημαίνει προστασία τους από τυχόν βλάβη ή αλλοίωση, η οποία θα προέλθει, είτε από κακόβουλη ενέργεια, είτε από βλάβη στο υπολογιστικό σύστημα που αποθηκεύονται τα δεδομένα.

- Τα δεδομένα από τη φύσή τους πρέπει να συσχετίζονται με **κλειδί συσχέτισης** ένα ή περισσότερα πεδία; Τα δεδομένα ενός συστήματος πολλές φορές είναι οργανωμένα σε οντότητες, που, όμως, δεν χρησιμοποιούνται ανεξάρτητα, αλλά σε σύνδεση με άλλες οντότητες.

  Για παράδειγμα, αν μια οντότητα αντιστοιχεί στους εργαζόμενους μιας επιχείρησης, ενώ μια άλλη στα τμήματα της επιχείρησης, στα οποία οι εργαζόμενοι είναι ενταγμένοι, υπάρχει πάντα η ανάγκη, οι πληροφορίες που αντιστοιχούν στις δύο αυτές οντότητες να εξάγονται σε συνδυασμό. Δηλαδή, όταν απαιτηθούν πληροφορίες για τους υπαλλήλους, να είναι δυνατό με μία αναζήτηση να αντληθούν πληροφορίες και για τα τμήματα στα οποία είναι ενταγμένοι.

- Τα προγράμματα είναι σε μεσοπρόθεσμη – μακροπρόθεσμη προοπτική **αμετάβλητα** ή απαιτούν συχνές μετατροπές; Δηλαδή, πρόκειται για ένα σύστημα, που για ένα σημαντικό εύρος χρόνου θα ικανοποιεί τις ανάγκες του πελάτη, χωρίς να χρειάζεται αλλαγές στα προγράμματα που το αποτελούν, ή σημαντικές αναδιοργανώσεις στη μορφή, με την οποία τα δεδομένα του είναι αποθηκευμένα;

Αν η απάντηση είναι καταφατική σε όλα τα προηγούμενα ερωτήματα (ενώ στο τελευταίο ερώ-

τημα η απάντηση είναι ότι τα δεδομένα υπόκεινται σε συχνές μεταβολές), τότε η επιλογή ενός συστήματος διαχείρισης βάσεων δεδομένων είναι η καταλληλότερη λύση, αφού μόνον έτσι διασφαλίζονται όλες οι παραπάνω απαιτήσεις. Σε όλες τις άλλες περιπτώσεις, στις οποίες περιλαμβάνονται επιστημονικά προβλήματα, στατιστικές εφαρμογές, εφαρμογές προσαρμοσμένες στις ανάγκες του διαδικτύου, εφαρμογές αυτοματισμού γραφείου, διαχείρισης λογιστηρίου, αποθήκης, παραγγελιών, γενικά εφαρμογές που διαχειρίζονται δεδομένα για τα οποία δεν απαιτείται η πολύπλοκη όσο και ευέλικτη οργάνωση που προσφέρουν τα σύγχρονα ΣΔΒΔ, το εργαλείο ανάπτυξης θα επιλεγεί ανάμεσα στα πολλά σύγχρονα περιβάλλοντα, που καλύπτουν όλων των ειδών τις ανάγκες.

Εφαρμογές γενικής χρήσης μπορούν να αναπτυχθούν με τις εξελιγμένες σύγχρονες εκδόσεις των γλωσσών 3ης γενιάς (π.χ Pascal, C, C++, Cobol κ.α.). Μάλιστα, επειδή για τις περισσότερες από τις γλώσσες αυτές υπάρχουν εκδόσεις κατάλληλες για τα πιο γνωστά λειτουργικά συστήματα Η/Υ, μια καλά σχεδιασμένη εφαρμογή είναι δυνατό να λειτουργεί εξίσου αποτελεσματικά σε διαφορετικού τύπου υπολογιστές, εξασφαλίζοντας έτσι καλύτερες προοπτικές αναβάθμισης, μεταφερσιμότητας και εμπορικής αξίας της εφαρμογής.

Παραθυρικά λέγονται τα περιβάλλοντα στα οποία οι εφαρμογές, αλλά και οι επιμέρους λειτουργίες τους εκτελούνται στην οθόνη μέσα σε ανεξάρτητα μεταξύ τους παράθυρα. Χαρακτηριστικό παραθυρικό περιβάλλον είναι το περιβάλλον που δημιουργούν τα MS-Windows.

Ο **χρόνος ανάπτυξης** είναι και αυτός ένας σοβαρός παράγοντας για την επιλογή του λογισμικού ανάπτυξης. Μια πληθώρα εργαλείων κάτω από το γενικό τίτλο "**συστήματα γρήγορης ανάπτυξης εφαρμογών**" (*rapid application development systems - RAD)* είναι διαθέσιμα για να εξυπηρετήσουν ακριβώς το σκοπό που ο τίτλος τους δηλώνει, δηλαδή την κατά το δυνατόν ταχύτερη ανάπτυξη εφαρμογών. Τα συστήματα αυτά συνήθως προσφέρουν μεγάλες δυνατότητες συνδεσιμότητας με τα δημοφιλέστερα ΣΔΒΔ, αποτελώντας ολοκληρωμένα περιβάλλοντα ανάπτυξης. Στην πλειοψηφία τους χρησιμοποιούνται για τη σχεδίαση εφαρμογών που λειτουργούν σε παραθυρικά περιβάλλοντα, αλλά με δυνατότητα διαχείρισης δεδομένων, τα οποία βρίσκονται, είτε στον ίδιο υπολογιστή στον οποίο "τρέχει" η εφαρμογή, είτε σε άλλον ακολουθώντας το πρότυπο πελάτη - εξυπηρετητή *(client-server)*. Γνωστά τέτοια εργαλεία είναι τα Delphi, C++ Builder, Visual Basic, PowerBuilder, Oracle Developer κ.α.

Για εξειδικευμένες επιστημονικές ή στατιστικές εφαρμογές πολλοί προτιμούν τη χρήση δημοφιλών πακέτων, τα οποία μπορούν να χρησιμοποιηθούν και από χρήστες που δεν είναι εξειδικευμένοι στον προγραμματισμό. Τα πακέτα αυτά, εκτός από ιδιαίτερα αποτελεσματικά για τις συγκεκριμένες εφαρμογές είναι και αρκετά φιλικά στη χρήση τους. Τα ίδια ισχύουν και για τα λεγόμενα πακέτα αυτοματισμού γραφείου. Γνωστές εφαρμογές που καλύπτουν κάποιες από τις κατηγορίες αυτές εφαρμογών είναι: (Autocad, CorelDraw, SPSS, SAS, Excel, Word, Lotus Notes κ.ο.κ).

Τέλος, για μια κατηγορία τυποποιημένων εφαρμογών που ικανοποιούν συγκεκριμένες ειδικές ανάγκες μεγάλου αριθμού χρηστών όπως διαχείριση αποθήκης, πελατολόγιο, μισθοδοσία κ.λ.π., γνωστές εταιρίες παραγωγής λογισμικού έχουν δημιουργήσει έτοιμες λύσεις, που μπορούν να προσαρμοστούν απόλυτα στις ανάγκες κάθε πελάτη. Οι έτοιμες αυτές λύσεις, που

έχουν πολύ μικρότερο κόστος, σε σχέση με τις κατά παραγγελία εφαρμογές, μπορεί να προσφέρουν ενδιαφέρουσες λύσεις στα προβλήματα αυτού του τύπου. **Βασικός παράγοντας στην επιλογή των κατάλληλων εφαρμογών είναι το κόστος αγοράς και υποστήριξης (συντήρησης) τους.**

Αν πελάτης για το σύστημα είναι μια μεγάλη εταιρία που διαθέτει τμήμα μηχανογράφησης, πρέπει να εξεταστεί ποια είναι η διαθεσιμότητα, η εμπειρία και οι δυνατότητες του προσωπικού του τμήματος. Αν οι υπόλοιποι παράγοντες επιλογής οδηγήσουν, για παράδειγμα, σε ένα σύστημα βάσης δεδομένων, τότε πρέπει να εξεταστεί κατά πόσο με κατάλληλη εκπαίδευση το προσωπικό αυτό είναι σε θέση να φέρει σε πέρας το έργο της ανάπτυξης του συστήματος, βασισμένο κύρια στις δικές του δυνάμεις. Παράλληλα πρέπει να εξεταστεί αν ο απαιτούμενος χρόνος εκπαίδευσης των υπαλλήλων είναι τέτοιος, που θα τους εμποδίσει να διεκπεραιώσουν τις τρέχουσες υποχρεώσεις τους καθώς και τι κόστος μπορεί να έχει αυτό για το εργασιακό τους περιβάλλον. Σε περιπτώσεις που ο πελάτης επιλέγει να παραγγείλει κάποια έτοιμη εφαρμογή σε τρίτους, πρέπει να εξετάσει σοβαρά το κόστος που θα επωμιστεί, καθώς και το αν τα αποτελέσματα, από άποψη αναμενόμενης θετικής επίδρασης στην εργασία του, είναι αντάξια αυτού του κόστους.

Σε ερωτήματα σαν αυτά απαντά σε μεγάλο βαθμό η διαδικασία της ανάλυσης του συστήματος. Όμως, αν η διαδικασία ανάλυσης οδηγήσει σε επιλογή ενός εργαλείου (π.χ. του RAD) για την υλοποίηση, θα πρέπει να επιλεγεί εκείνο το πακέτο της κατηγορίας αυτής, που σε σχέση με τους παράγοντες υπολογισμού του κόστους που αναφέρθηκαν, είναι περισσότερο συμφέρον. Αν συνεκτιμηθούν σωστά όλοι οι παράγοντες που αναφέρθηκαν, τότε είναι πολύ πιθανό, ότι θα οδηγήσει στις καλύτερες από κάθε άποψη επιλογές εργαλείων ανάπτυξης.

8.2. ΕΠΙΛΟΓΗ ΤΩΝ ΕΦΑΡΜΟΓΩΝ ΣΤΙΣ ΟΠΟΙΕΣ ΘΑ ΕΝΣΩΜΑΤΩΘΟΥΝ ΟΙ ΛΥΣΕΙΣ

Με βάση το σχεδιασμό της εφαρμογής και συνεκτιμώντας όλους τους παράγοντες που αναφέρθηκαν παραπάνω, γίνεται πλέον η επιλογή των εργαλείων ως εξής:

Σε περίπτωση που όλη η εφαρμογή υλοποιηθεί με χρήση **ενός και μόνο εργαλείου ανάπτυξης**, θα πρέπει να γίνει κατανομή της εργασίας παραγωγής του κώδικα στους προγραμματιστές. Την κατανομή κάνει εκείνος που έχει οριστεί υπεύθυνος για το έργο. Συνήθως, κάποιοι προγραμματιστές θα αναλάβουν τον κώδικα για τις **διαδικασίες της εφαρμογής**, κάποιοι άλλοι θα αναλάβουν τους **καταλόγους** (*menu*) **επιλογής διαδικασιών** και στο τέλος θα γίνει η σύνδεση του καταλόγου των επιλογών, με τις διαδικασίες.

Σε περίπτωση που η εφαρμογή απαιτεί τη χρησιμοποίηση παραπάνω του ενός εργαλεία ανάπτυξης, π.χ. η οργάνωση των δεδομένων υλοποιείται με ένα ΣΔΒΔ, ενώ τα προγράμματα διαχείρισής

τους με κάποιο εργαλείο τύπου RAD, η κατανομή των εργασιών μπορεί να γίνει περίπου έτσι:

Ομάδα 1: Δημιουργία των πινάκων δεδομένων, των συσχετίσεων τους και των περιορισμών ακεραιότητας των δεδομένων.

Ομάδα 2: Δημιουργία των καταλόγων επιλογής προγραμμάτων και διαδικασιών της εφαρμογής.

Ομάδα 3: Δημιουργία των φορμών εισαγωγής στοιχείων.

Ομάδα 4: Δημιουργία των εκτυπώσιμων αναφορών.

Ομάδα 5: Δημιουργία οθονών βοήθειας για τις επιμέρους λειτουργίες της εφαρμογής και των σχετικών εγχειριδίων βοήθειας, αν κάτι τέτοιο απαιτείται.

Ομάδα 6: Συντονισμός και συνεργασία με τις προηγούμενες ομάδες για τη σύνδεση των επιμέρους τμημάτων της εφαρμογής και έλεγχος καλής λειτουργίας και ποιότητας των προγραμμάτων. Επισήμανση τυχόν λαθών σχεδιασμού και αδυναμιών της εφαρμογής και συνεργασία με τις άλλες ομάδες, οι οποίες θα κάνουν τις απαραίτητες διορθώσεις.

Ομάδα 7: Δημιουγία φορμών ερωταπαντήσεων – αναζητήσεων (με τα περισσότερο ζητούμενα στοιχεία από τη Β.Δ., βάση των αναγκών που δημιουργήθηκε, αλλά και με τη δυνατότητα ο χρήστης να ζητά στοιχεία αλάζοντας τα κριτήρια – συνθήκες αναζήτησης).

8.3. ΔΙΕΡΕΥΝΗΣΗ ΤΗΣ ΔΥΝΑΤΟΤΗΤΑΣ ΣΥΝΕΡΓΑΣΙΑΣ ΚΑΙ ΑΝΤΑΛΛΑΓΗΣ ΔΕΔΟΜΕΝΩΝ ΜΕΤΑΞΥ ΔΙΑΦΟΡΕΤΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

Βασικό ζήτημα που εξετάζεται κατά τη φάση της ανάπτυξης, ειδικά δε των σύγχρονων συστημάτων, είναι κατά πόσο θα ενσωματωθούν κάποιες δυνατότητες ανταλλαγής δεδομένων και επικοινωνίας με άλλα συστήματα – εφαρμογές.

Η τάση που επικρατεί πλέον διεθνώς είναι η δημιουργία συστημάτων ανοικτών στην ανταλλαγή στοιχείων με άλλες εφαρμογές, οι οποίες είτε είναι εγκατεστημένες στα ίδια υπολογιστικά συστήματα, είτε σε άλλα, ενταγμένα στο ίδιο **τοπικό δίκτυο** (*Local Area Network - LAN*) ή σε διαφορετικά **δίκτυα ευρείας περιοχής** (*Wide Area Networks - WAN*). Άλλη δυνατότητα είναι στοιχεία από το σύστημα να διοχετεύονται στο **διαδίκτυο**, με δυνατότητα ευρείας ή περιορισμένης πρόσβασης, ανάλογα με τα επίπεδα ασφάλειας τα οποία έχουν θέσει για την πρόσβαση αυτή οι διαχειριστές του συστήματος.

Η απλούστερη δυνατότητα επικοινωνίας και ανταλλαγής στοιχείων μεταξύ συστημάτων (εφαρμογών) είναι με βάση τις τεχνολογίες **DDE** και **OLE**, που παρουσιάστηκαν συνοπτικά σε προηγούμενα κεφάλαια.

Επίσης, στην περίπτωση που τα προγράμματα εφαρμογών προσπελαύνουν δεδομένα μιας βάσης, με την οποία δεν συνδέονται με απευθείας οδηγούς – προγράμματα σύνδεσης, εναλλακτικό τρόπο σύνδεσης προσφέρουν κάποιες άλλες βιβλιοθήκες κώδικα, που παρέχουν τη

δυνατότητα αυτής της σύνδεσης και είναι γνωστές σαν **βιβλιοθήκες (οδηγοί) σύνδεσης βάσης δεδομένων με Java κώδικα** (*Java Database Connectivity - JDBC*) και **σύνδεσης αντικειμένων με βάσεις δεδομένων** (*Object Database Connectivity - ODBC*). Στην πρώτη περίπτωση (JDBC), η δημοφιλής, εξαιτίας της τεράστιας διάδοσης της χρήσης της για εφαρμογές με προορισμό το διαδίκτυο γλώσσα προγραμματισμού Java, επικοινωνεί με τις πιο δημοφιλείς σύγχρονες βάσεις δεδομένων με κώδικα ο οποίος δεν αλλάζει, ανεξάρτητα σε ποια βάση είναι οργανωμένα τα δεδομένα. Στη δεύτερη περίπτωση με χρήση των ODBC βιβλιοθηκών, περισσότερο παραδοσιακές γλώσσες προγραμματισμού ή RAD πακέτα επικοινωνούν επίσης με όλες τις δημοφιλείς βάσεις δεδομένων.

Παράλληλα παραμένει χρήσιμος και επίκαιρος ο τρόπος ανταλλαγής δεδομένων με χρήση απλών αρχείων κειμένου, χωρίς ειδικούς χαρακτήρες. Τα αρχεία αυτά είναι γνωστά σαν **αρχεία τύπου ASCII**, επειδή περιέχουν μόνο τους τυπικούς χαρακτήρες ASCII από το ομώνυμο σύνολο χαρακτήρων που χρησιμοποιούνται στους ηλεκτρονικούς υπολογιστές. Τα περισσότερα προγράμματα εφαρμογών προβλέπουν από την κατασκευή τους (ή μπορούν να επιτρέψουν την προσθήκη στο μέλλον), τη δυνατότητα ανταλλαγής στοιχείων με τον τρόπο αυτό.

Οι παραπάνω τρόποι, που αναλυτικότερα περιγράφηκαν στο κεφάλαιο 2, είναι οι γνωστότεροι τρόποι ανταλλαγής στοιχείων μεταξύ των σύγχρονων εφαρμογών Η/Υ. Ένα σύστημα θεωρείται καλά σχεδιασμένο, όταν ακόμα και αν δεν υπάρχει άμεση ανάγκη, αφήνει ένα παράθυρο επικοινωνίας με άλλα συστήματα, παράθυρο που ίσως χρειαστεί να ανοίξει κάποια στιγμή. Η ελάχιστη δυνατή πρόβλεψη ανταλλαγής στοιχείων είναι η μέθοδος εξαγωγής δεδομένων σε μορφή αρχείου ASCII.

Ανακεφαλαίωση / περίληψη

Ένας από τους σημαντικότερους παράγοντες επιτυχούς ανάπτυξης ενός συστήματος είναι η σωστή επιλογή των εργαλείων, με τα οποία θα αναπτυχθεί το σύστημα. Η επιλογή εξαρτάται από πολλές παραμέτρους και αποσκοπεί στο να βρεθεί το κατάλληλο από τεχνική, αλλά και από οικονομική άποψη εργαλείο.

Αν το σύστημα είναι κτισμένο με τη λογική της σύνθετης διαχείρισης δεδομένων, τότε η συνηθέστερη επιλογή είναι ένα ΣΔΒΔ. Σε άλλες περιπτώσεις μπορεί να γίνει επιλογή μίας περισσότερο παραδοσιακής λύσης, όπως είναι μια γλώσσα προγραμματισμού τρίτης γενιάς, ή ένα πιο σύγχρονο περιβάλλον τύπου RAD.

Σημαντική τέλος για την ποιότητα και την αξιοπιστία του συστήματος που θα παραχθεί με οποιοδήποτε εργαλείο, είναι η δυνατότητα ανταλλαγής δεδομένων του συστήματος με άλλες εφαρμογές. Ο πιο συνηθισμένος τρόπος επικοινωνίας και ανταλλαγής δεδομένων είναι μέσω απλών αρχείων ASCII, η και με πιο σύγχρονους τρόπους, όπως είναι οι τεχνικές DDE, OLE, JDBC, ODBC κ.λ.π.

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Αναφέρετε τους σημαντικότερους παράγοντες, που κατά τη γνώμη σας πρέπει να ληφθούν υπόψη για την επιλογή των καταλληλότερων εργαλείων ανάπτυξης ενός συστήματος.
2. Αναφέρετε τους πιο διαδεδομένους τρόπους επικοινωνίας και ανταλλαγής στοιχείων μεταξύ διαφορετικών εφαρμογών. Ποιες διαφορές υπάρχουν μεταξύ τους;

Γλωσσάριο - Ευρετήριο όρων

δίκτυα ευρείας περιοχής (WAN) *(Wide Area Networks)* - Συστήματα υπολογιστών διασυνδεδεμένα μεταξύ τους σε δίκτυα, τα οποία είναι εγκατεστημένα σε διαφορετικούς, απομακρυσμένους φυσικούς χώρους και χρησιμοποιούν δημόσιο μέσο σύνδεσης (π.χ., γραμμές ΟΤΕ) [178].
αρχεία τύπου ASCII - Αρχεία τα οποία αποτελούνται μόνο από εκτυπώσιμούς χαρακτήρες του κώδικα ASCII και είναι αναγνώσιμα από οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου [179].
οδηγοί σύνδεσης αντικειμένων με βάσεις δεδομένων - Βιβλιοθήκες κώδικα που επιτρέπουν τη σύνδεση διαφορετικών προγραμμάτων, τα οποία λειτουργούν σε περιβάλλον MS-Windows, με όλες τις δημοφιλείς βάσεις δεδομένων, έτσι ώστε να εξασφαλίζουν τη μέγιστη δυνατή μεταφερσιμότητα μεταξύ των εφαρμογών και την ανεξαρτησία τους από την επιλογή ΣΔΒΔ [179].
συστήματα γρήγορης ανάπτυξης εφαρμογών (RAD) *(Rapid Application Development Systems)* - Προγράμματα και εργαλεία που απλοποιούν και επιταχύνουν την ανάπτυξη σύγχρονων παραθυρικών εφαρμογών, κυρίως εφαρμογών που αξιοποιούν δεδομένα από ΣΔΒΔ [176].

τοπικά δίκτυα (LAN) *(Local Area Networks)* - Σε αντίθεση με τα δίκτυα ευρείας περιοχής πρόκειται για δίκτυα υπολογιστών οι οποίοι βρίσκονται σε κοντινές αποστάσεις μεταξύ τους π.χ. στο ίδιο γραφείο, στο ίδιο κτίριο ή σε κοντινά κτίρια του ίδιου οργανισμού και χρησιμοποιούν ιδιωτικό μέσο σύνδεσης (π.χ., σύρμα περιεστραμένο, ομοαξονικό καλώδιο κ.α.) [178].

Βιβλιογραφία

Χρήστου Π. Σκουρλά, "Σχεσιακές Βάσεις Δεδομένων", Εκδόσεις Νέων Τεχνολογιών, 1992.
Εμμ. Α. Γιακουμάκη, "Τεχνολογία Λογισμικού", Εκδόσεις Α. Σταμούλη, 1994, Τόμοι Α' και Β'.

ΚΕΦΑΛΑΙΟ 9

ΤΜΗΜΑΤΙΚΗ ΥΛΟΠΟΙΗΣΗ ΣΤΟ ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ ΠΟΥ ΕΧΕΙ ΕΠΙΛΕΓΕΙ

ΠΕΡΙΕΧΟΜΕΝΑ

9.1. Καθορισμός του παραδοτέου κάθε εργασίας

9.2. Υλοποίηση

ΕΙΣΑΓΩΓΗ

Όπως έχει αναφερθεί ήδη από το κεφάλαιο 4, το πρόβλημα και κατ' επέκταση η εφαρμογή, διαχωρίζεται σε επιμέρους τμήματα, καθένα από τα οποία μπορεί να υλοποιηθεί χωριστά.

- Ποια, όμως, είναι τα τμήματα στα οποία διασπάται μια εφαρμογή;
- Και πώς εξασφαλίζεται ότι τα τμήματα αυτά θα μπορέσουν να ενωθούν χωρίς πρόβλημα σε ενιαία εφαρμογή;

Τυπικά τμήματα στα οποία διασπάται μια εφαρμογή λογισμικού είναι τα τμήματα κώδικα (προγράμματος) που υλοποιούν τις φόρμες εισαγωγής δεδομένων, τους καταλόγους επιλογών (μενού), τις αναφορές *(reports)*, τα ερωταπαντήματα *(queries)* κ.λ.π. Κάθε τμήμα μπορεί να ανατεθεί σε διαφορετικό προγραμματιστή ή ομάδα προγραμματιστών. Ιδιαίτερα σημαντικό είναι να προσδιοριστούν με ακρίβεια τα παραδοτέα κάθε ομάδας, ώστε να είναι δυνατή στη συνέχεια η σύνθεσή τους σε ενιαίο πρόγραμμα, απαλλαγμένο λογικών σφαλμάτων, επικαλύψεων, κενών και αντιφάσεων.

Η παραγωγή υψηλής ποιότητας λογισμικού απαιτεί η προγραμματιστική ομάδα να έχει ένα καλά καθορισμένο σχέδιο (πλάνο) του έργου υλοποίησης, μια καλά οργανωμένη συγκρότηση σε ανθρώπινους πόρους και μια συνολική συνειδητοποίηση των καθηκόντων και των ευθυνών κάθε μέλους της ομάδας. Η ομάδα υλοποίησης θα πρέπει να έχει στη διάθεση της καλά ορισμένες απαιτήσεις του λογισμικού, προδιαγραφές αρχιτεκτονικού σχεδίου και μια λεπτομερή περιγραφή σχεδίασης. Επίσης κάθε μέλος της ομάδας θα πρέπει να κατανοεί τους στόχους και τις γενικές αρχές της υλοποίησης. Όμως, όλα τα παραπάνω εξαρτώνται σε μεγάλο βαθμό και από το πρότυπο (κύκλο) ανάπτυξης λογισμικού, που έχει υιοθετηθεί για την ανάπτυξη της εφαρμογής.

Στο παρόν κεφάλαιο θα περιγραφούν οι βασικές αρχές που διέπουν την υλοποίηση σύμφωνα με τους γνωστότερους κύκλους ανάπτυξης λογισμικού καθώς και ορισμένες βασικές τεχνικές ανάπτυξης αξιόπιστου λογισμικού. Τέλος, θα δοθεί ένα εκτενές και αρκετά αναλυτικό παράδειγμα τμηματικής υλοποίησης λογισμικού, προκειμένου να γίνουν κατανοητές οι απαιτούμενες προς τούτο εργασίες.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να γνωρίσει ο μαθητής τα πλεονεκτήματα και μειονεκτήματα των γνωστότερων προτύπων (κύκλων) ανάπτυξης λογισμικού, καθώς και τις βασικές αρχές που διέπουν την υλοποίηση στο καθένα απ' αυτά.
2. Να μπορεί να διασπά την υλοποίηση σε τμήματα και να καθορίζει τα παραδοτέα κάθε επιμέρους εργασίας.
3. Να μπορεί να αξιολογεί, να παραδίδει και να προσαρμόζει τα προϊόντα της υλοποίησης.
4. Να μπορεί να εφαρμόζει τις αρχές προγραμματισμού με αξιοπιστία κατά την ανάπτυξη του λογισμικού.

9.1. ΚΑΘΟΡΙΣΜΟΣ ΤΟΥ ΠΑΡΑΔΟΤΕΟΥ ΚΑΘΕ ΕΠΙΜΕΡΟΥΣ ΕΡΓΑΣΙΑΣ

ΟΡΙΣΜΟΣ Με τον όρο "**υλοποίηση**" εννοούμε τη διαδικασία μετατροπής του σχεδίου του λογισμικού σε ένα τελικό προϊόν, που ικανοποιεί τις προδιαγραφές του.

Το μεγαλύτερο μέρος υλοποιήσεων λογισμικού αναπτύσσεται σε περιβάλλοντα στα οποία ομάδες προγραμματιστών ή συλλογές ομάδων, εμπλέκονται στο να υλοποιήσουν το προϊόν λογισμικού. Οι κύκλοι ανάπτυξης του λογισμικού περιγράφουν τη διαδικασία που χρησιμοποιείται για να παραχθεί το λογισμικό. Οι κυριότεροι κύκλοι ανάπτυξης λογισμικού είναι οι εξής:

Η **κυκλική** *(cyclic)* ανάπτυξη λογισμικού **(σχήμα 9.1)** αρχίζει με την σύλληψη μιας ιδέας για ένα έργο ή προϊόν και, εφόσον διερευνηθεί η εφικτότητα υλοποίησης, γίνεται καθορισμός των προδιαγραφών οι οποίες μπορεί και να διατυπωθούν σε ένα συμβόλαιο για το έργο, εφόσον αυτό είναι επιθυμητό. Ακολουθεί σχεδίαση και υλοποίηση των προγραμμάτων, πραγματοποιούνται έλεγχοι ορθότητας του λογισμικού και διερευνάται η δυνατότητα αποδοχής του. Αν το λογισμικό δε γίνει αποδεκτό επιβάλλεται αναθεώρηση του, γεγονός που οδηγεί σε επανάληψη της κυκλικής ανάπτυξης μέχρι το προϊόν λογισμικού να αποδειχθεί ότι πληροί τα κριτήρια αποδοχής. Εφόσον είναι επιθυμητό, μπορεί να πραγματοποιηθεί "διαχείριση της διαδικασίας ανάπτυξης", η οποία ασκεί εποπτεία για την τήρηση των χρονοδιαγραμμάτων υλοποίησης, την οικονομική διαχείριση και την διαχείριση οικονομικών, ανθρωπίνων και υλικών πόρων.

Σχήμα 9.1 - Στρατηγική ανάπτυξης λογισμικού.

Το **πρότυπο του καταρράκτη** *(waterfall)* **(σχήμα 9.2)** έχει περιγραφεί στο Κεφάλαιο 7. Περιλαμβάνει τη συντήρηση σα βασικό τμήμα του κύκλου ανάπτυξης του λογισμικού και δεν απαιτεί την ύπαρξη σταδίου διαχείρισης της διαδικασίας ανάπτυξης λογισμικού.

Σχήμα 9.2. - Στρατηγική ανάπτυξης λογισμικού με το πρότυπο του καταρράκτη.

Σχήμα 9.3. - Στρατηγική ανάπτυξης λογισμικού με χρήση του ευμετάβλητου κύκλου ανάπτυξης.

Στον **ευμετάβλητο** *(evolutionary)* **(σχήμα 9.3)** κύκλο ανάπτυξης λογισμικού το περιβάλλον και οι προδιαγραφές, που διαρκώς μεταβάλλονται, αποτελούν βασικές συνιστώσες και πολλά τμηματικά προγραμματιστικά προϊόντα παράγονται σταδιακά, προκειμένου να ικανοποιήσουν τις αλλαγές των προδιαγραφών, συγκλίνοντας προς ένα τελικό προϊόν λογισμικού.

Οι βασικές διαδικασίες που τηρούνται κατά την υλοποίηση με βάση το παραπάνω πρότυπο είναι οι εξής:

- **Η παράδοση** *(delivery)* κάποιου τμήματος του λογισμικού σε έναν πραγματικό τελικό χρήστη,
- **η μέτρηση** *(measurement)* της αξίας του λογισμικού από τον χρήστη, με βάση τη φιλικότητα του και την ικανοποίηση λειτουργικών απαιτήσεων του χρήστη και
- **η προσαρμογή** *(adjustment)* του λογισμικού τόσο στη σχεδίαση όσο και στους στόχους, με βάση τα ρεαλιστικά δεδομένα και αποτελέσματα, που επιβάλλει η αναθεωρημένη πορεία ανάπτυξης, σύμφωνα με τους αναθεωρημένους στόχους.

Στον **κύκλο ανάπτυξης V** *(V cycle)* **(σχήμα 9.4)** εφαρμόζεται **επιβεβαίωση της ποιότητας του λογισμικού** *(software quality assurance)* σε κάθε στάδιο και το έργο ανάπτυξης του λογισμικού αντιμετωπίζεται σαν ολότητα. Επιβεβαίωση της ποιότητας του λογισμικού ονομάζεται η διαδικασία που ακολουθείται για να εξασφαλιστεί ότι ένα σύστημα λογισμικού ικανοποιεί τις προδιαγραφές του και καλύπτει τις απαιτήσεις των μελλοντικών χρηστών. Παράλληλα με την υλοποίηση του λογισμικού πραγματοποιούνται έλεγχοι των προϊόντων της υλοποίησης, με στόχο τη δημιουργία ενός ολοκληρωμένου και υψηλής ποιότητας προϊόντος, το οποίο έχει δημιουργηθεί με τρόπο ώστε να διευκολύνεται η συντήρηση του.

Σχήμα 9.4. - Στρατηγική ανάπτυξης λογισμικού με χρήση του μοντέλου V.

Ανεξάρτητα από το πρότυπο ανάπτυξης, είναι απαραίτητο να καθορίζονται τα **παραδοτέα** *(deliverables)* σε κάθε στάδιο του κύκλου ανάπτυξης του λογισμικού.

Ενδεικτικά παραδοτέα είναι τα εξής:

1. Υλοποιήσεις λειτουργιών **διαχείρισης αρχείων** *(File Management):* Αφορούν στο σύνολο των διαδικασιών για τη δημιουργία, τη φύλαξη και την ενημέρωση των αρχείων.
2. Υλοποιήσεις **διεπαφών** *(interfaces):* Αφορούν στη σύνδεση συστημάτων, συσκευών (εκτυπωτών, μονάδων ταινίας κ.λ.π.) ή προγραμμάτων με διαφορετικά ή παρόμοια χαρακτηριστικά.
3. Υλοποιήσεις **οθονών** *(forms):* Αφορούν στις οθόνες διαλογικής επικοινωνίας του χρήστη με την εφαρμογή λογισμικού. Στις οθόνες αυτές προτυπώνονται ορισμένα στοιχεία που υποδεικνύουν τις θέσεις που απαιτείται να συμπληρωθούν με συγκεκριμένες πληροφορίες εισόδου προς τον υπολογιστή.
4. Υλοποιήσεις **ερωταπαντημάτων** *(queries)* . Αφορούν στη διατύπωση απλών ή σύνθετων κριτηρίων αναζήτησης εγγραφών, που βρίσκονται αποθηκευμένες σε ένα ή περισσότερα αρχεία της εφαρμογής λογισμικού.
5. Υλοποιήσεις **αναφορών** *(reports):* Παρουσιάζουν τα αποτελέσματα της επεξεργασίας δεδομένων από την εφαρμογή λογισμικού στην οθόνη ή στον εκτυπωτή. Συνήθως οι αναφορές χρησιμοποιούν τα αποτελέσματα αναζητήσεων ή ερωταπαντημάτων, τα οποία παρουσιάζουν μορφοποιημένα με τον τρόπο που επιθυμεί ο προγραμματιστής.
6. Υλοποιήσεις των **καταλόγων (μενού) επιλογών** *(menus)* της εφαρμογής.

Παράδειγμα: Στη συνέχεια παρατίθεται ένα παράδειγμα καθορισμού των παραδοτέων σε ένα πρόβλημα ανάπτυξης λογισμικού εμπορικών εφαρμογών.

Καθορισμός του προβλήματος: Μία Βιβλιοθήκη ενδιαφέρεται για ένα σύστημα διαχείρισης της διαδικασίας δανεισμού των βιβλίων της.

Όνομα στοιχείου δεδομένων	Συμβολικό όνομα
Κωδικός βιβλίου	KODIKOS-BIBLIOY
Συγγραφέας	SYGRAFEAS
Τίτλος βιβλίου	TITLOS
Η διαθέσιμη, την τρέχουσα χρονική στιγμή, ποσότητα στη βιβλιοθήκη από το συγκεκριμένο βιβλίο	POSOTHTA

Για κάθε βιβλίο στη βιβλιοθήκη τηρούνται τα παρακάτω στοιχεία:

Όνομα στοιχείου δεδομένων	Συμβολικό όνομα
Κωδικός βιβλίου	KODIKOS-BIBLIOY
Κωδικός δανειζόμενου	KODIKOS-DANEIZOMENOY
Επώνυμο δανειζόμενου	EPONYMO-DANEIZOMENOY
Όνομα δανειζόμενου	ONOMA-DANEIZOMENOY
Γενικά στοιχεία δανειζόμενου	GENERAL-DANEIZOMENOY
Ημερομηνία επιστροφής βιβλίου	HMEROMHNIA-EPISTROFIS
Κατάσταση της δοσοληψίας που μπορεί να έχει τρεις τιμές: -"0" όταν η δοσοληψία έχει ολοκληρωθεί και το βιβλίο έχει επιστραφεί -"1" όταν το βιβλίο είναι δανεισμένο αλλά δεν έχει εκπνεύσει η περίοδος δανεισμού -"2" όταν το βιβλίο είναι δανεισμένο αλλά έχει εκπνεύσει η περίοδος δανεισμού	KATASTASH

Επίσης, για κάθε δοσοληψία δανεισμού βιβλίου τηρούνται τα παρακάτω στοιχεία:
Να σχεδιάσετε και να υλοποιήσετε μία εφαρμογή η οποία να επιτελεί τις παρακάτω λειτουργίες:
1. Να διαχειρίζεται το Αρχείο Βιβλίων

	Όνομα	Πεδία
Αρχείο Βιβλίων	BIBLIA	KODIKOS-BIBLIOY, SYGRAFEAS, TITLOS, POSOTHTA

(α) να δημιουργεί το Αρχείο Βιβλίων:

(β) να επιτρέπει τις παρακάτω πράξεις διαχείρισης στο Αρχείο Βιβλίων
- εισαγωγή στοιχείων
- διαγραφή στοιχείων
- ενημέρωση στοιχείων
- ανάκτηση στοιχείων

2. Να υλοποιεί τη δοσοληψία δανεισμού ενός βιβλίου. Το πρόγραμμα θα δέχεται από το τερματικό τον κωδικό του βιβλίου, τον κωδικό δανειζομένου, το επώνυμο και το όνομα του δανειζομένου καθώς και την ημερομηνία επιστροφής, και θα μεριμνά για την ορθή ενημέρωση των Αρχείων Βιβλίων και Δοσοληψιών Δανεισμού. Ο κάθε συνδρομητής μπορεί να δανειστεί ταυτόχρονα το πολύ μέχρι πέντε βιβλία και μόνο ένα αντίγραφο του ίδιου βιβλίου. Ο έλεγχος για την τήρηση αυτού του περιορισμού είναι ευθύνη του προγράμματος.

Η μορφή του αρχείου Δοσοληψιών Δανεισμού είναι η ακόλουθη:

Αρχείο Δοσοληψιών Δανεισμού	Όνομα	Πεδία
	DANEISMOS	KODIKOS-BIBLIOY, KODIKOS-DANEIZOMENOY, EPONYMO-DANEIZOMENOY, ONOMA-DANEIZOMENOY, GENERAL-DANEIZOMENOY, HMEROMHNIA-EPISTROFIS, KATASTASH

3. Να παράγει κατάσταση με τους συνδρομητές που έχουν δανειστεί το βιβλίο με συγκεκριμένο κωδικό ταξινομημένους ως προς το ονοματεπώνυμο. Η κατάσταση να έχει τη μορφή:

ΚΩΔΙΚΟΣ ΒΙΒΛΙΟΥ : ########

ΕΠΩΝΥΜΟ ΟΝΟΜΑ ΗΜΕΡΟΜΗΝΙΑ ΕΠΙΣΤΡΟΦΗΣ

4. Να παράγει κατάσταση που θα περιλαμβάνει τους τίτλους και τους κωδικούς των βιβλίων που πρέπει να επιστραφούν μια συγκεκριμένη ημερομηνία που επιθυμεί ο χρήστης της εφαρμογής, καθώς επίσης και τον ολικό αριθμό των βιβλίων που αναμένεται να επιστραφούν την ημερομηνία αυτή. Η κατάσταση να περιλαμβάνει κατάλληλες επικεφαλίδες.

5. Να δίνει στο χρήστη τη δυνατότητα να εκτελεί μια επιλογή που να ενημερώνει το Αρχείο Δοσοληψιών για την κατάσταση κάθε δοσοληψίας, δηλ. για τα τυχόν βιβλία που έχουν καθυστερήσει να επιστραφούν μέχρι την τρέχουσα ημέρα.

6. Να παράγει κατάσταση με τα βιβλία, η επιστροφή των οποίων έχει καθυστερήσει και για τα οποία δεν υπάρχει αντίτυπο στη βιβλιοθήκη.

Επιπλέον **απαιτήσεις** που έχουν προσδιοριστεί κατά τη φάση της σχεδίασης του λογισμικού είναι οι εξής:

- Η τεκμηρίωση της σχεδίασης να γίνει μέσω διαγραμμάτων HIPO και συνοδευτικών IPO.
- Να χρησιμοποιηθούν τα IPO για την τεκμηρίωση του κώδικα.
- Η διεπαφή με το χρήστη να είναι διαλογική (μέσω χρήσης συστήματος οθονών για πρόσβαση στις διάφορες λειτουργίες της) και φιλική (μέσω παροχής on-line βοήθειας).
- Η ασφάλεια των δεδομένων της εφαρμογής να εξασφαλιστεί μέσω αυτοματοποιημένων διαδικασιών, που θα είναι διαθέσιμες από το γενικό μενού επιλογών της εφαρμογής και θα παρέχουν δυνατότητες:
  (ι) δημιουργίας αντιγράφων ασφαλείας *(backup)*
  (ιι) επαναφόρτωσης αντιγράφων ασφαλείας *(restore)*
  (ιιι) αναδιοργάνωσης των ευρετηρίων *(rebuild)*
  (ιν) χρήσης κωδικών ασφαλείας *(passwords)* για την πρόσβαση στα διάφορα τμήματα της εφαρμογής.

Σχέδιο (πλάνο) του έργου

1. Λαμβάνοντας υπόψη τις γραμμογραφήσεις των αρχείων να ορίσετε τα ευρετήρια, όπως περιγράφονται στην εκφώνηση.
2. Η εφαρμογή πρέπει να έχει τις παρακάτω επιλογές:

1. Διαχείριση Αρχείου Βιβλίων	1.1 Εισαγωγή 1.2 Διαγραφή 1.3 Αναζήτηση 1.4 Διόρθωση
2. Δοσοληψία Δανεισμού	
3. Ενημέρωση Κατάστασης Αρχείου Δοσοληψιών	
4. Καταστάσεις	4.1 Αλφαβητική κατάσταση δανειζόμενων βιβλίων 4.2 Κατάσταση επιστρεφομένων βιβλίων 4.3 Κατάσταση καθυστερημένων βιβλίων
5. Βοηθητικά προγράμματα	5.1 Δημιουργία αντιγράφων ασφαλείας 5.2 Επαναφόρτωση αντιγράφων ασφαλείας 5.3 Αναδιοργάνωση Ευρετηρίων 5.4 Σύστημα κωδικών ασφαλείας
0. Έξοδος	

όπου η αντιστοιχία επιλογών και λειτουργιών περιγράφεται από τον παρακάτω πίνακα:

Επιλογή	Λειτουργία
1.1 Διαχείριση αρχείου Βιβλίων	Εισαγωγή, διαγραφή, αναζήτηση και διόρθωση των εγγραφών του αρχείου BIBLIA
2. Δοσοληψία δανεισμού	Το πρόγραμμα θα δέχεται από το τερματικό τον κωδικό του βιβλίου, τον κωδικό δανειζόμενου, το επώνυμο και το όνομα του δανειζόμενου καθώς και την ημερομηνία επιστροφής, και θα μεριμνά για την ορθή ενημέρωση των Αρχείων BIBLIA και DANEISMOS. Ο κάθε συνδρομητής μπορεί να δανειστεί ταυτόχρονα το πολύ μέχρι πέντε βιβλία.
3. Ενημέρωση Κατάστασης Αρχείου Δοσοληψιών	Ενημέρωση του πεδίου KATASTASH του αρχείου DANEISMOS με την κατάσταση κάθε δοσοληψίας, δηλ. για τα τυχόν βιβλία που έχουν καθυστερήσει να επιστραφούν μέχρι την τρέχουσα ημέρα.
4.1. Αλφαβητική κατάσταση δανειζομένων βιβλίων	Κατάσταση με τους συνδρομητές που έχουν δανειστεί το βιβλίο με συγκεκριμένο κωδικό x ταξινομημένους ως προς το ονοματεπώνυμο
4.2. Κατάσταση επιστρεφομένων βιβλίων	Κατάσταση με τους τίτλους και τους κωδικούς των βιβλίων που πρέπει να επιστραφούν μια συγκεκριμένη ημερομηνία x, καθώς και με τον ολικό αριθμό των βιβλίων που αναμένεται να επιστραφούν την ημερομηνία x.
4.3. Καθυστερημένα βιβλία	Κατάσταση με τα βιβλία η επιστροφή των οποίων έχει καθυστερήσει και για τα οποία δεν υπάρχει αντίτυπο στη βιβλιοθήκη

3. Οι περιγραφές των πεδίων των αρχείων είναι:

Όνομα πεδίου	Τύπος	Πλάτος
KODIKOS-BIBLIOY	Ακολουθία χαρακτήρων	8
SYGRAFEAS	Ακολουθία χαρακτήρων	40
TITLOS	Ακολουθία χαρακτήρων	50
POSOTHTA	Ακέραιος	2
KODIKOS-DANEIZOMENOY	Ακολουθία χαρακτήρων	10
EPONYMO-DANEIZOMENOY	Ακολουθία χαρακτήρων	20
ONOMA-DANEIZOMENOY	Ακολουθία χαρακτήρων	20
GENERAL-DANEIZOMENOY	Ακολουθία χαρακτήρων	100
HMEROMHNIA-EPISTROFIS	Ημερομηνία	8
KATASTASH	Ακολουθία χαρακτήρων	1

Προδιαγραφές υλοποίησης της εφαρμογής

Για την επίλυση της άσκησης θα θεωρήσουμε ότι η οργάνωση όλων των αρχείων είναι σειριακή με δείκτες *(indexed sequential)* και ισχύουν τα παρακάτω πρωτεύοντα και δευτερεύοντα κλειδιά:

Όνομα Αρχείου	Πρωτεύον Κλειδί	Δευτερεύοντα Κλειδιά
BIBLIA	KWDIKOS-BIBLIOY	Σύνθετο:{EPONYMO-DANEIZOMENOY, ONOMA-DANEIZOMENOY}
DANEISMOS	Σύνθετο:{KODIKOS-BIBLIOY, KODIKOS-DANEIZOMENOY}	KATASTASH

1. Η δημιουργία των αρχείων και των ευρετηρίων τους να γίνει προγραμματιστικά.
2. Οι λειτουργίες στα αρχεία να γίνουν μέσω ανάπτυξης προγράμματος με τις διαδικασίες των πράξεων για κάθε αρχείο ξεχωριστά.
3. Η παραγωγή καταστάσεων να γίνει προγραμματιστικά (ένα πρόγραμμα για κάθε κατάσταση).
4. Τα μενού επιλογών της εφαρμογής να κατασκευαστούν προγραμματιστικά, μέσω χρήσης διαδικασιών και συναρτήσεων κατασκευής οθονών.

ΠΑΡΑΤΗΡΗΣΗ

Η διαδικασία που ακολουθήθηκε παραπάνω για τον καθορισμό των παραδοτέων δεν είναι απαραίτητα ο μοναδικός τρόπος αντιμετώπισης του οποιουδήποτε προβλήματος. Είναι δυνατό κάποιος να επιλέξει διαφορετική πορεία. Σε κάθε περίπτωση όμως, τα παραδοτέα που θα σχεδιασθούν πρέπει να υλοποιηθούν με μια λογική σειρά και να επιτρέπουν τον καλύτερο δυνατό καταμερισμό εργασιών.

Παράδειγμα:

Στη συνέχεια παρουσιάζεται ένα μικρό παράδειγμα με παραδοτέα τις υλοποιήσεις **ερωταπαντημάτων** *(queries)*.

Ας θεωρήσουμε ότι μια επιχείρηση ενδιαφέρεται για μια εφαρμογή λογισμικού διαχείρισης των υπαλλήλων της με στόχο να μπορεί να παράγει αποτελέσματα αναζητήσεων που να απαντούν στα κάτωθι ερώτημα:

➢ Προσδιορισμός υφισταμένων κάθε υπαλλήλου που είναι προϊστάμενος.
➢ Προσδιορισμός των υπαλλήλων που έχουν μισθό μεγαλύτερο των 200.000 δραχμών.

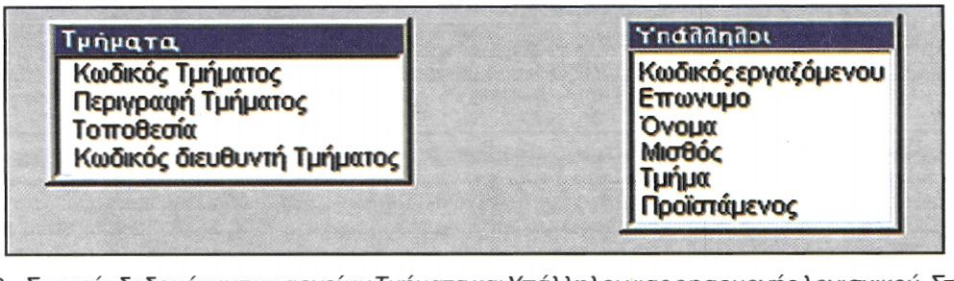
Ένα δείγμα αποτελεσμάτων του ζητούμενου ερωταπαντήματος παρουσιάζεται στο **σχήμα 9.5.**

Υφιστάμενοι

Προϊστάμενος	6000		
Κωδικός εργαζομένου	Επώνυμο	Όνομα	Μισθός
0001	Αντωνίου	Βασίλης	500.000,00 Δρχ.
0010	Γαλάνης	Δημήτρης	200.000,00 Δρχ.
0011	Γυπαράκης	Γεώργιος	250.000,00 Δρχ.
0012	Αθανασίου	Τίμος	210.000,00 Δρχ.
0100	Τζιβαρίδης	Νίκος	300.000,00 Δρχ.
Προϊστάμενος	6100		
Κωδικός εργαζομένου	Επώνυμο	Όνομα	Μισθός
0000	Κοροβήλος	Βλάσσης	150.000,00 Δρχ.

Σχήμα 9.5. - Δείγμα αναφοράς υφισταμένων κάθε υπαλλήλου που είναι προϊστάμενος

Είναι προφανές ότι η εφαρμογή λογισμικού θα πρέπει να διαχειρίζεται το αρχείο υπαλλήλων και το αρχείο τμημάτων τα οποία μπορούν να έχουν ως πεδία τα στοιχεία δεδομένων του παρακάτω **σχήματος 9.6.**



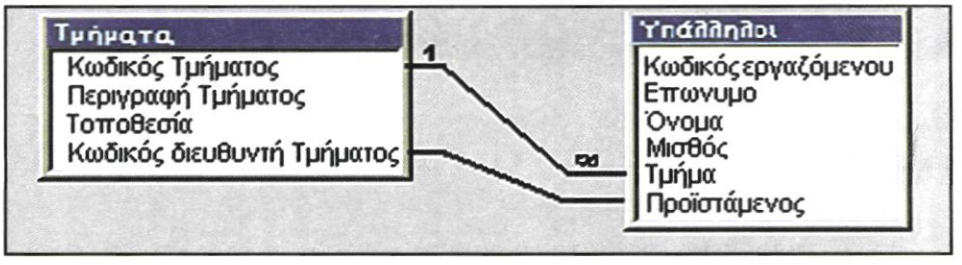
Σχήμα 9.6. - Στοιχεία δεδομένων των αρχείων Τμήματα και Υπάλληλοι μιας εφαρμογής λογισμικού. Στα στοιχεία δεδομένων "Κωδικός διευθυντή τμήματος" του αρχείου "Τμήματα" και "Προϊστάμενος" του αρχείου "Υπάλληλοι" θα πρέπει να καταχωρηθεί τιμή που έχει δοθεί στο στοιχείο δεδομένων "Κωδικός Εργαζομένου".

Προκειμένου να παράγουμε τα ζητούμενα ερωταπαντήματαθα πρέπει να επιλέξουμε από κάθε αρχείο τα απαραίτητα στοιχεία δεδομένων τα οποία είναι **πεδία κλειδιά,** καθώς επίσης και **τα πεδία συσχέτισης** (βλέπε κεφάλαιο 6).

Παράδειγμα:

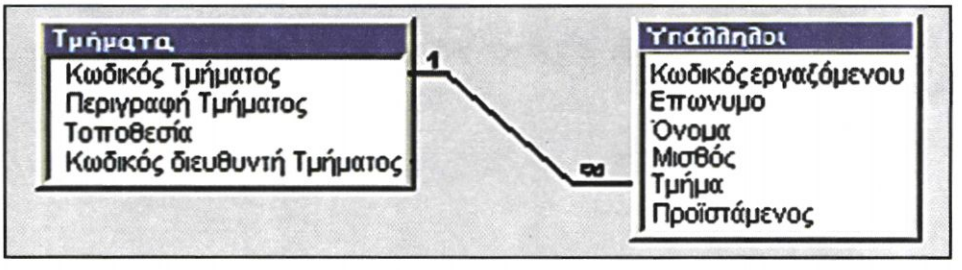
Το ερωταπάντημα προσδιορισμού των υφισταμένων κάθε υπαλλήλου που είναι προϊστάμενος γίνεται με χρήση των πεδίων συσχέτισης που παρουσιάζονται στο **σχήμα 9.7.** με στόχο να περιέχει από το αρχείο "Υπάλληλοι" τις τιμές των πεδίων" Προϊστάμενος", "Κωδικός εργαζομένου", "Επώνυμο"," Όνομα", "Μισθός".

Για την επίλυση του ερωταπαντήματος προσδιορισμού υπαλλήλων με μισθό μεγαλύτερο των διακοσίων χιλιάδων δραχμών αρκεί να χρησιμοποιηθούν τα πεδία συσχέτισης που παρουσιά-



Σχήμα 9.7. - Ερωταπάντημα προσδιορισμού των υφισταμένων κάθε υπαλλήλου που είναι προϊστάμενος. Το πεδίο κλειδί του αρχείου Υπάλληλοι είναι ο "Κωδικός εργαζομένου" ο οποίος είναι μοναδικός για κάθε υπάλληλο. Τα πεδία συσχέτισης των δύο αρχείων είναι αυτά τα οποία είναι συνδεδεμένα με γραμμές.

ζονται στο **σχήμα 9.8** και κριτήριο αναζήτησης το στοιχείο δεδομένων" Μισθό"ς του αρχείου "Υπάλληλοι" να έχει τιμή μεγαλύτερη ή ίση των διακοσίων χιλιάδων.



Σχήμα 9.8. - Πεδία συσχέτισης ερωταπαντήματος υπαλλήλων με μισθό μεγαλύτερο των διακοσίων χιλιάδων δραχμών και με περιεχόμενα τον Κωδικό εργαζομένου, το Επώνυμο, το Όνομα και την Περιγραφή Τμήματος ταξινομημένη ως προς το Επώνυμο και το Όνομα.

Σημείωση:

Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 9, Μάθημα 1ο, Δραστηριότητα 1.

9.2. ΥΛΟΠΟΙΗΣΗ

Η φάση υλοποίησης του λογισμικού έχει σαν στόχο τη μεταφορά των προδιαγραφών της σχεδίασης του λογισμικού σε πηγαίο κώδικα *(source code)*, με τη χρήση κάποιας γλώσσας προγραμματισμού. Ο πρωταρχικός στόχος της υλοποίησης είναι να γραφεί πηγαίος κώδικας και τεκμηρίωση, έτσι ώστε η προσαρμογή του κώδικα, ανάλογα με τις προδιαγραφές που έχουν τεθεί, να μπορεί εύκολα να ελεγχθεί, να γίνει δηλαδή εκσφαλμάτωση. Αυτός ο στόχος μπορεί να επιτευχθεί γράφοντας πηγαίο κώδικα όσο το δυνατόν περισσότερο δομημένο και καθαρό. Η απλότητα, η διαύγεια και η κομψότητα είναι τα χαρακτηριστικά των καλών προγραμμάτων. Η σκοτεινότητα και η πολυπλοκότητα είναι ενδείξεις ανεπαρκούς σχεδιασμού και όχι ευθείας σκέψης.

Οι δυσκολίες που αντιμετωπίζονται κατά τη διάρκεια της υλοποίησης είναι αποτέλεσμα του γεγονότος ότι ο προγραμματιστής διεξάγει ταυτόχρονα ανάλυση, σχεδιασμό και προγραμματισμό. Αυτό είναι αναπόφευκτο, όταν για διάφορους λόγους δεν είναι δυνατή η υλοποίηση, όπως προδιαγράφεται στη φάση της σχεδίασης. Τότε ο προγραμματιστής επανεξετάζει τα δεδομένα της ανάλυσης και κατά περίπτωση τροποποιεί το σχέδιο της εφαρμογής, ώστε αυτό να γίνει υλοποιήσιμο.

Για την υλοποίηση συχνά επιλέγονται γλώσσες προγραμματισμού προσανατολισμένες στον προγραμματιστή, δηλ. γλώσσες που χρησιμοποιούν λέξεις και σύνταξη που πλησιάζει σε κάποια ομιλούμενη γλώσσα (αγγλική). Οι γλώσσες αυτές ονομάζονται **υψηλού επιπέδου** *(high level programming languages)*. Επίσης σημαντική θεωρείται η υιοθέτηση από τον προγραμματιστή της τεχνικής του δομημένου προγραμματισμού.

Ο **δομημένος προγραμματισμός** *(structured programming)* είναι συστηματική μεθοδολογία για το σχεδιασμό προγραμμάτων με προσέγγιση "από το γενικό στο ειδικό". Με τη μεθοδολογία αυτή δημιουργούνται ενότητες προγραμμάτων τα οποία μπορούν να δοκιμαστούν ανεξάρτητα ενώ διευκολύνεται η κατανόηση του συνολικού προγράμματος.

Ορισμός

Τέλος, μεγάλη προσοχή πρέπει να δοθεί στη **σωστή επιλογή τύπων δεδομένων.** Οι τύποι δεδομένων είναι σύνολα δεδομένων που χαρακτηρίζονται και διαφοροποιούνται από άλλα σύνολα δεδομένων, με βάση την αναπαράσταση τους στην κύρια μνήμη του υπολογιστή και τις πράξεις που μπορούν να εκτελεστούν με αυτά. (π.χ. ακέραιος, πραγματικός κ.λ.π.).

Τα **προϊόντα** της υλοποίησης μπορεί να είναι:

- **Συστήματα Εφαρμογών** *(application systems)*: Ένα σύστημα εφαρμογών είναι σύνολο προγραμμάτων για την εκτέλεση όλων των λειτουργιών ενός συστήματος (π.χ. πρόγραμμα βιβλιοθήκης).
- **Υποσυστήματα** *(subsystems)*: Σύνολο προγραμμάτων για την εκτέλεση των λειτουργιών κάθε υποσυστήματος ενός συστήματος. Υποσύστημα ονομάζεται ένα αυτοτελές τμήμα ενός συστήματος το οποίο εκτελεί μια από τις βασικές του λειτουργίες.

- **Μονάδες** *(modules):* Αυτοτελή τμήματα ενός συστήματος ή υποσυστήματος ή προγράμματος που λειτουργούν ανεξάρτητα ή συνεργάζονται με άλλα τμήματα.
- **Ενότητες** *(units):* Προγράμματα που εκτελούν μια συγκεκριμένη εργασία η οποία αποτελείται από ένα σύνολο λειτουργιών.
- **Συναρτήσεις** *(functions):* Τμήματα προγράμματος (ονομάζονται και υποπρογράμματα) που δέχονται παραμέτρους εισόδου και επιστρέφουν μόνο μία τιμή εξόδου.
- **Διαδικασίες** *(procedures):* Τμήματα προγράμματος (ονομάζονται και υποπρογράμματα) που δέχονται παραμέτρους εισόδου και επιστρέφουν πολλές τιμές εξόδου.

9.2.1. Προγραμματισμός με αξιοπιστία

Ο προγραμματισμός με αξιοπιστία αφορά στη χρήση προγραμματιστικών τεχνικών, με τις οποίες εξασφαλίζεται η ανοχή λαθών *(fault tolerance)* και η αποφυγή λαθών *(fault avoidance)*.

Ορισμός **Λάθος** *(Fault)* ονομάζεται κάθε προγραμματιστική ατέλεια που έχει σαν αποτέλεσμα την αδυναμία του προγράμματος να λειτουργήσει σύμφωνα με τις προδιαγραφές του.

Ορισμός **Ανοχή Λαθών** *(Fault Tolerance)* ονομάζεται η ικανότητα ενός συστήματος να εκτελεί σωστά συγκεκριμένους αλγορίθμους ανεξάρτητα από βλάβες του υλικού ή σφάλματα του λογισμικού. Ένα πρόγραμμα ονομάζεται ανεκτικό σε λάθη εάν εξακολουθεί να λειτουργεί παρά την ύπαρξη βλάβης ή σφάλματος σε κάποιο τμήμα του.

Δύο από τις πλέον καθιερωμένες τεχνικές για προγραμματισμό με αξιοπιστία είναι η τεχνική του αμυντικού προγραμματισμού και η τεχνική του προγραμματισμού πολλαπλών εκδόσεων.

Ορισμός **Αμυντικός Προγραμματισμός** *(defensive programming)* ονομάζεται, η προγραμματιστική προσέγγιση κατά την ανάπτυξη προγραμμάτων σύμφωνα με την οποία ενδεχόμενες ασυνέπειες που μπορεί να υπάρχουν στα προγράμματα πρέπει να προβλεφθούν πριν προκύψουν.

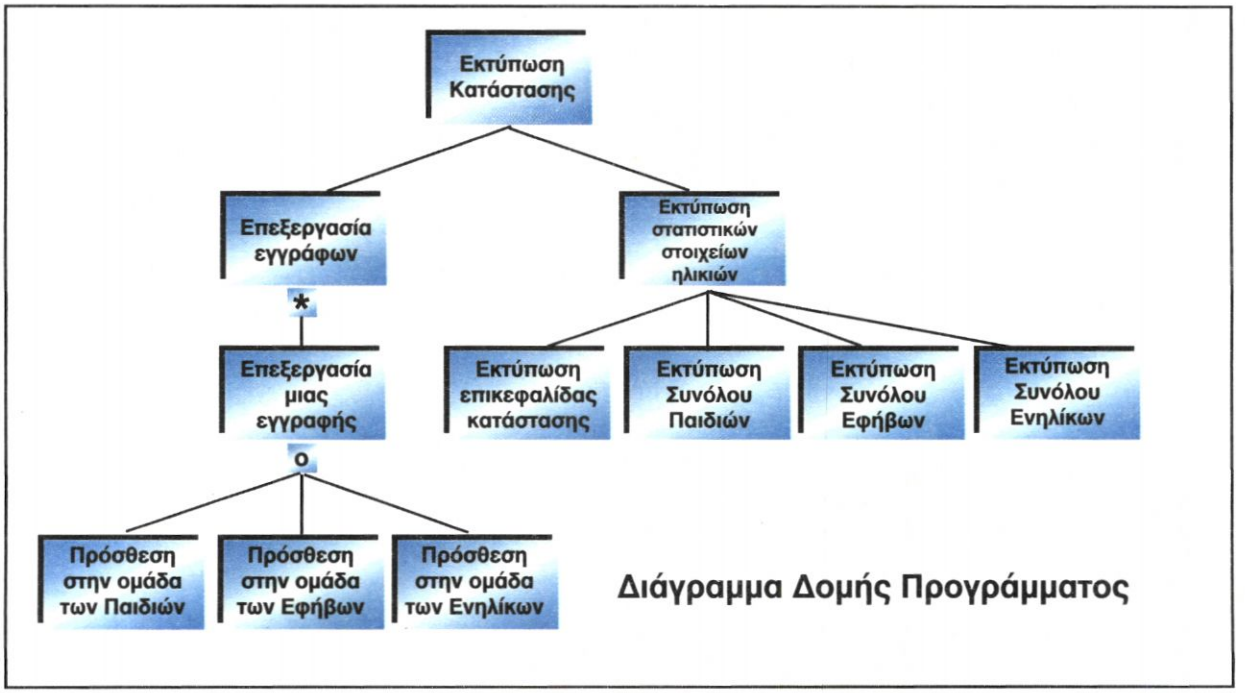
Αυτό επιτυγχάνεται με τα ακόλουθα βήματα:

1. Διαίρεση των αποφάσεων υλοποίησης σε χαμηλότερα επίπεδα.
2. Εντοπισμός των θεμάτων υλοποίησης, που δεν είναι πραγματικά ανεξάρτητα.
3. Προσεκτική απόδειξη ότι κάθε επιτυχές βήμα στον επαναπροσδιορισμό της προγραμματιστικής λειτουργίας αποτελεί μια πιστή επέκταση των προηγούμενων βημάτων.

Για να γίνει κατανοητή η τεχνική του αμυντικού προγραμματισμού παραθέτουμε ένα παράδειγμα.

Παράδειγμα: Έστω ότι ο προγραμματιστικός μας στόχος είναι η παραγωγή μιας κατάστασης στον εκτυπωτή, η οποία να περιέχει το σύνολο των παιδιών, το σύνολο των εφήβων και το σύνολο των ενηλίκων ενός Δήμου της χώρας μας, μέσω επεξεργασίας ενός αρχείου που περιέχει την ηλικία και το ονοματεπώνυμο των δημοτών. Στη διατύπωση του προβλήματος δια-

κρίνονται δύο αποφάσεις υλοποίησης. Η πρώτη αφορά στην επεξεργασία του αρχείου και η δεύτερη στην εκτύπωση των στατιστικών στοιχείων των ηλικιών. Κάθε απόφαση υλοποίησης προσεγγίζεται με τον ίδιο τρόπο και το αποτέλεσμα παρουσιάζεται στο **διάγραμμα δομής προγράμματος** *(program structure diagram)* του παρακάτω **σχήματος 9.9.**

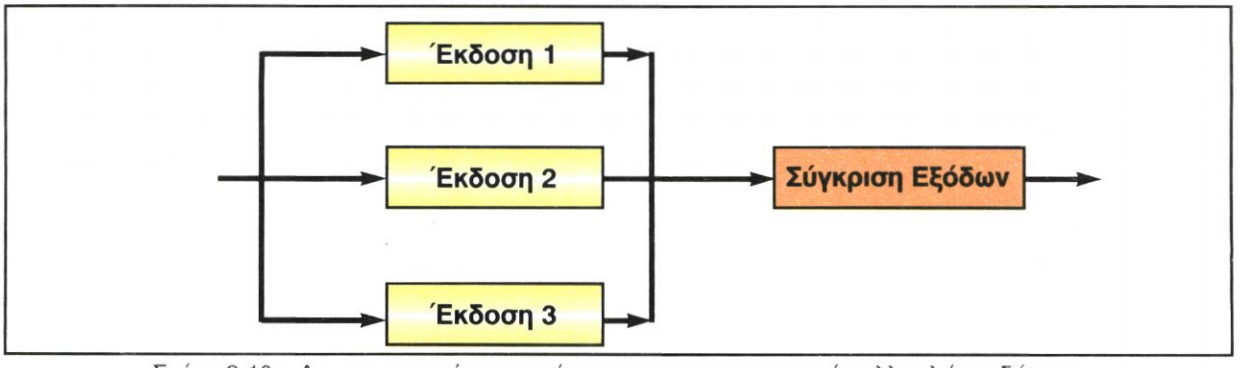


Σχήμα 9.9. - Διάγραμμα δομής προγράμματος που κατασκευάστηκε με εφαρμογή των αρχών του Αμυντικού Προγραμματισμού. Ο συμβολισμός "" δηλώνει επανάληψη και το "ο" επιλογή.*

Σημείωση:

Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 9, Μάθημα 2ο, Δραστηριότητα 2.

Μία άλλη καθιερωμένη τεχνική για προγραμματισμό με αξιοπιστία είναι η τεχνική του προγραμματισμού πολλαπλών εκδόσεων. Στον Προγραμματισμό Πολλαπλών Εκδόσεων *(N-version programming)* **(σχήμα 9.10)** το λογισμικό υλοποιείται σε έναν αριθμό εκδόσεων από διαφορετικές ομάδες και αυτές οι εκδόσεις εκτελούνται παράλληλα. Οι έξοδοι τους συγκρίνονται χρησιμοποιώντας σύστημα ψηφοφορίας και οι έξοδοι που δεν συμφωνούν απορρίπτονται. Το ελάχιστο πρέπει να υπάρχουν τρεις κωδικοποιήσεις του συστήματος.



Σχήμα 9.10. - Διαγραμματική αναπαράσταση προγραμματισμού πολλαπλών εκδόσεων

Η αποφυγή λαθών και συνεπώς η ανάπτυξη λογισμικού χωρίς λάθη στηρίζεται:

- Στον ακριβή ορισμό του συστήματος
- Στην ανάπτυξη λογισμικού βασισμένου σε απόκρυψη πληροφοριών *(information hiding)*. Η απόκρυψη πληροφορίας είναι τεχνική σχεδίασης συστημάτων λογισμικού σύμφωνα με την οποία το σχεδιαζόμενο σύστημα χωρίζεται σε προγραμματιστικές ενότητες. Κάθε ενότητα "κρύβει το δικό της μυστικό", δηλ. τις μορφές των δομών δεδομένων που περιέχει και τον τρόπο που χειρίζεται τα δεδομένα της. (υπενθυμίζουμε ότι οι δομές δεδομένων *(data structures)* είναι σύνολα δεδομένων μαζί με το σύνολο των επιτρεπτών σε αυτά πράξεων και λειτουργιών). Η τεχνική της απόκρυψης πληροφοριών διαθέτει επίσης τους απαραίτητους μηχανισμούς για την προσπέλαση και τη χρήση της από τις άλλες ενότητες, χωρίς να επιτρέπει την πρόσβαση στις εσωτερικές λειτουργίες της και τον απευθείας χειρισμό των δεδομένων της.
- Στην εκτεταμένη σύνταξη ανασκοπήσεων *(reviews)* ελέγχου κατά την διαδικασία της ανάπτυξης. Οι ανασκοπήσεις ελέγχου αφορούν στην εκτέλεση ενός προγράμματος με δοκιμαστικά δεδομένα και σύγκριση των αποτελεσμάτων του με τα αποτελέσματα υπολογισμών που έγιναν με το χέρι ή με τα αποτελέσματα άλλης δοκιμασμένης μεθόδου.
- Στον προσεκτικό σχεδιασμό του ελέγχου του συστήματος για να βρεθούν τα λάθη τα οποία δεν είχαν εντοπιστεί κατά τη διαδικασία της σύνταξης των αναφορών ελέγχου.

Σημείωση:

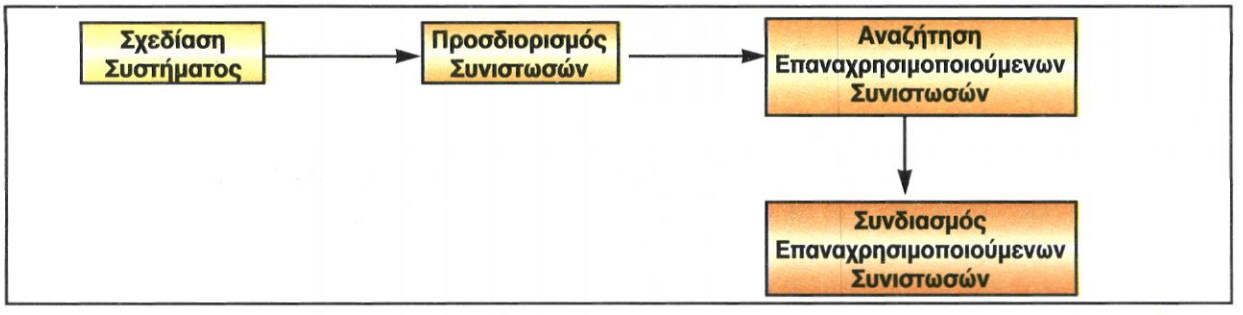
Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 9, Μάθημα 3ο, Δραστηριότητα 3 .

Κατά την υλοποίηση των εφαρμογών λογισμικού τα τελευταία χρόνια έχουν αρχίσει να υιοθετούνται τεχνικές προγραμματισμού, που επιτρέπουν την επαναχρησιμοποίηση του λογισμικού, δηλ., την ανάπτυξη εφαρμογών, που χρησιμοποιούν έτοιμα τμήματα κώδικα από άλλες εφαρμογές.

Η επαναχρησιμοποίηση λογισμικού *(Software Reuse)* προϋποθέτει την ύπαρξη μιας κατάλληλα οργανωμένης και τεκμηριωμένης συλλογής υλοποιημένων και προγραμματιστικά αξιόπιστων συνιστωσών. Οι συλλογές αυτές βρίσκονται σε βιβλιοθήκες λογισμικού *(software libraries)* οι οποίες περιέχουν ως συνιστώσες:

- Συστήματα Εφαρμογών
- Υποσυστήματα
- Μονάδες
- Ενότητες
- Συναρτήσεις
- Διαδικασίες

Στο **σχήμα 9.11** απεικονίζεται διαγραμματικά η διαδικασία ανάπτυξης λογισμικού με την τεχνική της επαναχρησιμοποίησης.



Σχήμα 9.11. - Ανάπτυξη Λογισμικού με επαναχρησιμοποίηση.

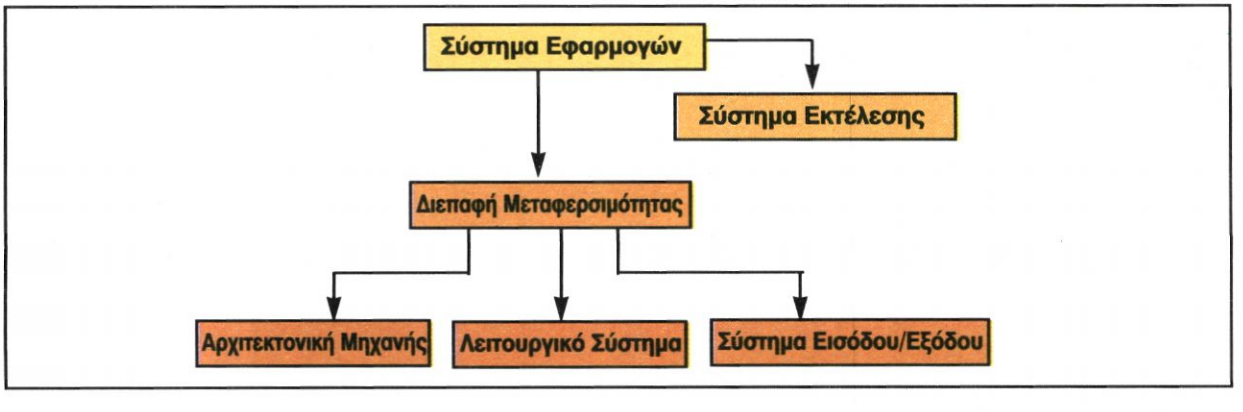
Υπάρχουν τρεις απαιτήσεις στην ανάπτυξη λογισμικού με επαναχρησιμοποίηση:
- Πρέπει να είναι δυνατόν να βρεθούν κατάλληλες επαναχρησιμοποιούμενες συνιστώσες.
- Αυτός που επαναχρησιμοποιεί τις συνιστώσες πρέπει να είναι ικανός να τις καταλάβει.
- Οι συνιστώσες πρέπει να διαθέτουν τεκμηρίωση, που να εξηγεί το πως μπορούν να επαναχρησιμοποιηθούν.

Μία ειδική περίπτωση επαναχρησιμοποίησης λογισμικού είναι η περίπτωση όπου ένα σύστημα εφαρμογών απαιτείται να υλοποιηθεί και να εκτελεστεί σε ένα εύρος διαφορετικών υπολογιστών και λειτουργικών συστημάτων.

· Μεταφορά *(Transportation)* Ονομάζεται η μετακίνηση του κώδικα του προγράμματος και των δεδομένων του από ένα περιβάλλον σε άλλο.

· Προσαρμογή *(Adaption)* Ονομάζεται το σύνολο των αλλαγών που απαιτούνται σε ένα πρόγραμμα για να δουλέψει σε ένα άλλο υπολογιστικό περιβάλλον.

Η γενική προσέγγιση που πρέπει να εφαρμοστεί για να υλοποιηθεί ένα μεταφέρσιμο σύστημα είναι να απομονώσουμε τα μέρη του συστήματος που εξαρτώνται από το εξωτερικό περιβάλλον σε μία διεπαφή μεταφερσιμότητας *(portability interface)*. Ολες οι λειτουργίες που κάνουν χρήση μη μεταφέρσιμων χαρακτηριστικών πρέπει να τοποθετηθούν σε αυτήν την διεπαφή. Στο **σχήμα 9.8.** απεικονίζεται η σχέση της διεπαφής μεταφερσιμότητας με τα λοιπά υποσυστήματα ενός υπολογιστικού περιβάλλοντος.



Σχήμα 9.8. - Η ενότητα μεταφερσιμότητας συστήματος εφαρμογών

Προβλήματα μεταφερσιμότητας παρουσιάζονται επειδή διαφορετικές αρχιτεκτονικές υπολογιστών επιβάλλουν διαφορετικούς τρόπους για την αναπαράσταση πληροφοριών στον υπολογιστή και διότι τα διαφορετικά λειτουργικά συστήματα παρέχουν διαφορετικά σύνολα κλήσεων συστήματος για την επίτευξη λειτουργιών, όπως είναι η διαχείριση διεργασιών και αρχείων.

Σημείωση: Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 9, Μάθημα 3ο, Δραστηριότητα 4.

9.2.2. Περίγραμμα Τεκμηρίωσης

Η Τεκμηρίωση *(software documentation)* μιας εφαρμογής λογισμικού περιλαμβάνει τον πηγαίο κώδικα για το σύστημα και όλα τα συνοδευτικά βοηθητικά κείμενα που φτιάχτηκαν κατά τη διάρκεια της ανάλυσης - σχεδίασης - υλοποίησης - δοκιμής και συντήρησης του συστήματος. Η τεκμηρίωση του κώδικα αποτελείται συνήθως από την εμφύτευση των ΙΡΟ της διαδικασίας σχεδίασης σε κάθε υποπρόγραμμα και από επεξηγηματικά σχόλια στις εντολές.

Οι προδιαγραφές απαιτήσεων, τα σχέδια ελέγχων, τα εγχειρίδια χρηστών, οι οδηγίες εγκατάστασης και οι οδηγίες συντήρησης αποτελούν παραδείγματα εγγράφων υποστήριξης *(supporting documents)*. Αυτά τα έγγραφα είναι προϊόντα της συστηματικής ανάπτυξης και συντήρησης του λογισμικού.

Μια σημαντική προσέγγιση στην ανάπτυξη λογισμικού εξασφαλίζεται μέσω ταυτόχρονης ανάπτυξης των εγγράφων υποστήριξης. Η διαθεσιμότητα τους είναι άμεση οποτεδήποτε απαιτηθεί. Η καλή συγκρότηση ενός πακέτου λογισμικού χαρακτηρίζεται από την ποιότητα, την ποσότητα και την χρησιμότητα των παρεχομένων εγγράφων υποστήριξης.

Σημείωση: Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 9, Μάθημα 4ο, Δραστηριότητα 5.

Ανακεφαλαίωση / περίληψη

Η υλοποίηση, όπως και ο σχεδιασμός, έχει σαν πρωταρχικό στόχο την παραγωγή λογισμικού, το οποίο να είναι εύκολα κατανοητό. Οι αλληλένδετες προγραμματιστικές δομές (συναρτήσεις, διαδικασίες, ενότητες, μονάδες, υποσυστήματα, συστήματα) πρέπει να ελέγχονται και ένα ευθύ προγραμματιστικό στυλ θα πρέπει να υιοθετείται. Η διαύγεια του πηγαίου κώδικα επιτυγχάνεται μέσω χρήσης δομημένων προγραμματιστικών τεχνικών, με καλό προγραμματιστικό στυλ, με κατάλληλες οδηγίες υποστήριξης, με καλά εσωτερικά σχόλια και με αξιοποίηση των δυνατοτήτων που παρέχονται από τις μοντέρνες γλώσσες προγραμματισμού. Η λεπτομερής περιγραφή του τρόπου λειτουργίας ενός πακέτου λογισμικού μέσω των εγγράφων υποστήριξης, σε συνδυασμό με αναλυτικό κατάλογο του πρωτογενούς προγράμματος αποτελεί μέτρο αξιολόγησης της ποιότητας ενός πακέτου λογισμικού.

Γλωσσάριο - Ευρετήριο όρων

βιβλιοθήκη λογισμικού *(software librarie)* - Συλλογή προγραμμάτων που που χρησιμοποιούνται συχνα [198].
δομές δεδομένων *(data structures)* - Είναι σύνολα δεδομένων μαζί με το σύνολο των επιτρεπτών σε αυτά πράξεων και λειτουργιών [198].
δομημένος προγραμματισμός *(structured programming)* - Προγραμματισμός κατά τον οποίο, αρχικά ορίζεται/σχεδιάζεται συνολικά το πρόγραμμα, στη συνέχεια οι μονάδες, οι ενότητες, οι διαδικασίες και οι συναρτήσεις και στο τέλος υλοποιούνται σε πηγαίο κώδικα οι λεπτομέρειες του [195].
τεκμηρίωση λογισμικού *(software documentation)* - Η πληροφορία που έχει σχεδιαστεί με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν [200].
υλοποίηση *(implementation)* - Η διαδικασία της μετατροπής του σχεδίου του λογισμικού σε ένα τελικό προϊόν που ικανοποιεί τις προδιαγραφές του [185].

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Ποιόν τρόπο ανάπτυξης λογισμικού θα επιλέγατε για την ανάπτυξη λογισμικού της Βιβλιοθήκης; Αιτιολογείστε την απάντηση σας.
2. Ποια είναι τα στάδια ανάπτυξης ενός προγράμματος;
3. Πως εξασφαλίζεται ο προγραμματισμός με αξιοπιστία;
4. Δημιουργείστε οθόνες για την επιλογή Διαχείρισης του αρχείου Βιβλίων της Βιβλιοθήκης.
5. Υλοποιείστε τις πράξεις Διαχείρισης Βιβλίων της εφαρμογής Βιβλιοθήκης.
6. Να παράγετε κατάσταση (αναφορά) με τους συνδρομητές της Βιβλιοθήκης που έχουν δανειστεί ένα βιβλίο με συγκεκριμένο κωδικό ταξινομημένους ως προς το ονοματεπώνυμο. Η κατάσταση να έχει τη μορφή:

ΚΩΔΙΚΟΣ ΒΙΒΛΙΟΥ : ########
ΕΠΩΝΥΜΟ ΟΝΟΜΑ ΗΜΕΡΟΜΗΝΙΑ ΕΠΙΣΤΡΟΦΗΣ

7. Να περιγράψετε τα έγγραφα τεκμηρίωσης της Εφαρμογής Βιβλιοθήκης που χρησιμοποιήσατε κατά την ανάπτυξη του λογισμικού σας.

Βιβλιογραφία

Sommerville, "Software Engineering", Addison-Wesley, 1985

Fairley R., "SoftwareEngineering Consepts", McGraw-Hill, 1985

Stevens W., "Using Structured Design: How to Make Programs Simple", Changeable, Flexible and Reusable, Wiley-Interscience, 1981

Π. Γαρίδης, Μ. Δεληγιαννάκης, "Σύγχρονο λεξικό Πληροφορικής", Εκδόσεις Δίαυλος, 1989

ΚΕΦΑΛΑΙΟ 10

ΣΥΝΘΕΣΗ ΤΩΝ ΕΠΙΜΕΡΟΥΣ ΕΡΓΑΣΙΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

10.1. Σύνθεση από πάνω προς τα κάτω
10.2. Επαλήθευση και επικύρωση
10.3. Σχεδίαση και διενέργεια ελέγχων

ΕΙΣΑΓΩΓΗ

Στο προηγούμενο κεφάλαιο είδαμε πώς είναι δυνατό να υλοποιηθεί τμηματικά μια εφαρμογή λογισμικού. Το ερώτημα που τίθεται στη συνέχεια είναι:

- Πώς πρέπει να προχωρήσουμε στη σύνθεση των παραδοτέων προϊόντων των επιμέρους εργασιών, ώστε να δημιουργήσουμε μια ολοκληρωμένη εφαρμογή, απαλλαγμένη σφαλμάτων και αντιφάσεων μεταξύ των μερών που θα την απαρτίζουν;

Για γίνει δυνατή η σύνθεση των παραδοτέων προϊόντων των επιμέρους εργασιών, που περιγράφηκαν στο προηγούμενο κεφάλαιο, είναι αναγκαίο να ανατρέξουμε στα δομικά στοιχεία (δηλ. τις μονάδες *(units)*) των χαμηλότερων επιπέδων του σχεδίου της εφαρμογής και να μελετήσουμε τα λεπτομερή τους χαρακτηριστικά, που περιγράφονται αναλυτικά σε αυτά τα επίπεδα. Έτσι, ξεκινώντας από τα χαμηλότερα ιεραρχικά επίπεδα προχωρούμε στα ανώτερα, όπου συναντούμε τις μεγάλες ενότητες (τμήματα), στις οποίες έχει διασπαστεί η εφαρμογή, και τις οποίες είδαμε σε προηγούμενο κεφάλαιο. Η μέθοδος αυτή σύνθεσης μιας εφαρμογής ονομάζεται σύνθεση "από κάτω προς τα πάνω".

Ο λεπτομερής σχεδιασμός, που αναλυτικά περιγράφηκε στο κεφάλαιο 7, αποτελεί το ενδιάμεσο στάδιο ανάμεσα στα "χαμηλότερα επίπεδα σχεδίασης" *(low level design)* και την υλοποίηση.

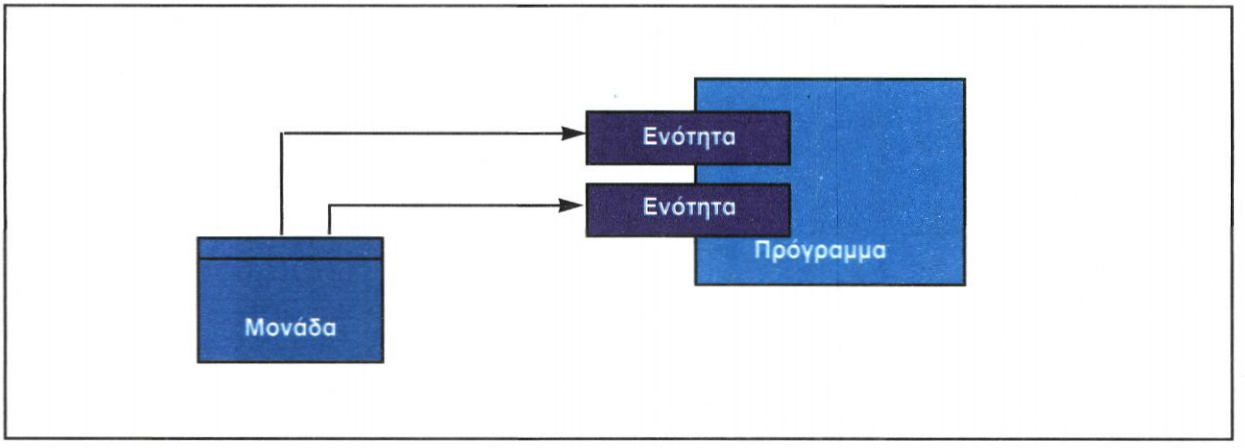
- Ένας επαρκής και λεπτομερής σχεδιασμός ελαχιστοποιεί τον αριθμό των ανεπιθύμητων εκπλήξεων κατά τη διάρκεια της υλοποίησης του προγράμματος.
- Η σύνθεση του λογισμικού (δηλ. η σύνδεση των επιμέρους τμημάτων του) είναι ένα βασικό θέμα του λεπτομερούς σχεδιασμού.
- Τέλος, ο σχεδιασμός του ελέγχου περιλαμβάνει τη διατύπωση των σκοπών του ελέγχου, την επιλογή των κριτηρίων ολοκλήρωσης, το σχεδιασμό της επέκτασης, την επιλογή συγκεκριμένων εργαλείων και τεχνικών που θα χρησιμοποιηθούν και περιγραφή δοκιμαστικών δεδομένων, σε συνδυασμό με προσδοκώμενα αποτελέσματα.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να υλοποιήσει μια εφαρμογή λογισμικού εφαρμόζοντας σε απλά προβλήματα πρότυπα (μοντέλα) οργάνωσης, παρουσίασης και σύνθεσης των δεδομένων
2. Να μπορεί να βιώσει την εμπειρία συμμετοχής σε ομάδες σε όλα τα στάδια ανάπτυξης προγράμματος
3. Να μπορεί συνθέτει τα πιο απλά μοντέλα υλοποίησης σε ένα πιο σύνθετο και ολοκληρωμένο
4. Να μπορεί να σχεδιάζει και να εφαρμόζει ελέγχους ενοτήτων και μονάδων

10.1. ΣΥΝΘΕΣΗ ΑΠΟ ΚΑΤΩ ΠΡΟΣ ΤΑ ΠΑΝΩ

Η **ολοκληρωμένη σύνθεση "από κάτω προς τα πάνω"** *(integrated bottom up development)* είναι μια ευρέως χρησιμοποιούμενη τεχνική, που βελτιώνει το σχεδιασμό, την υλοποίηση και τον έλεγχο της εφαρμογής. Χρησιμοποιώντας τη σύνθεση "από κάτω προς τα πάνω", η υλοποίηση ξεκινά από τα χαμηλότερα επίπεδα ενοτήτων. Τα επίπεδα αυτά έχουν πρωταρχική σημασία για τη δυνατότητα συντονισμού και συγχρονισμού των υψηλότερου επιπέδου ενοτήτων ή μονάδων. Οι **μονάδες** *(modules)* αποτελούν υλοποιήσεις αρχικών άλλων προγραμματιστικών **ενοτήτων** *(units)*, όπως φαίνεται και στο **σχήμα 10.1.**



Σχήμα 10.1. - Διαγραμματική αναπαράσταση μονάδας.

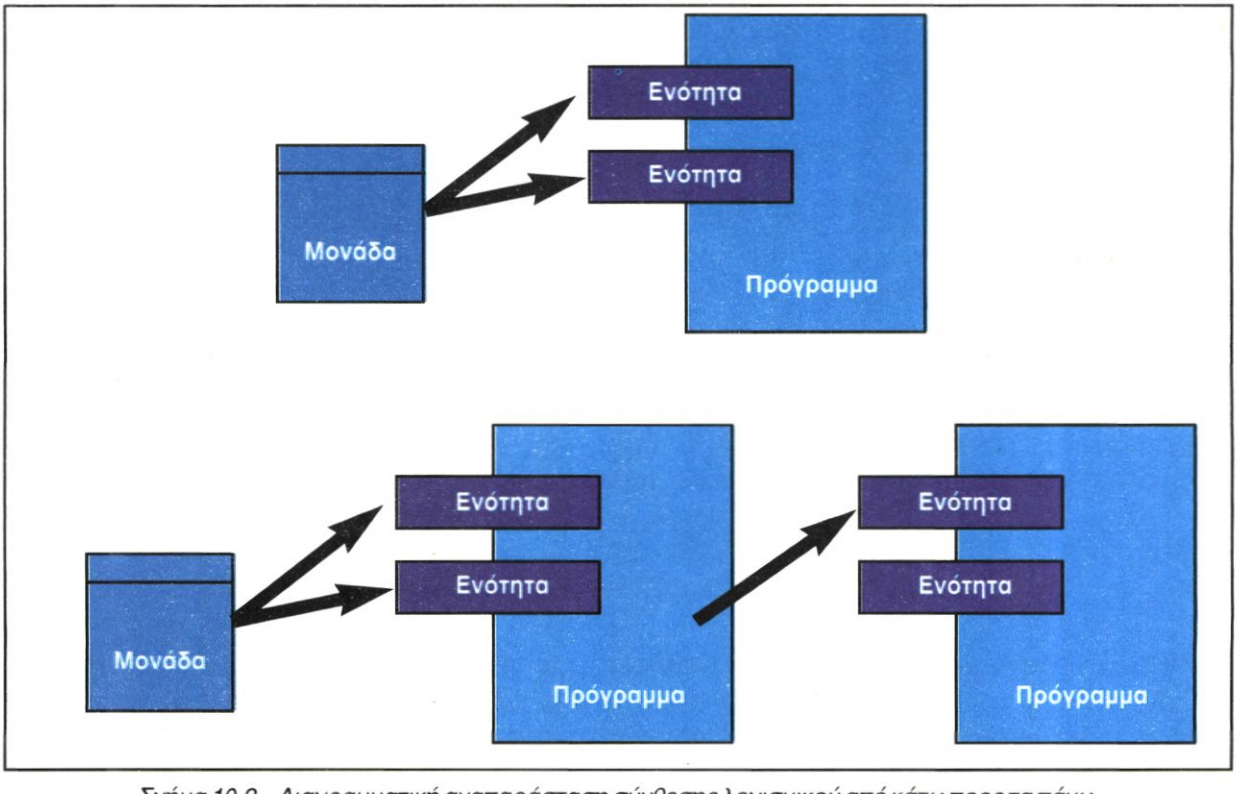
Γι αυτό συνήθως υπάρχει μια ιεραρχική δομή στη σχεδίαση του λογισμικού για ένα σύστημα, όπου οι διαδικασίες μπορούν να καλούν χαμηλότερου επιπέδου διαδικασίες, αλλά δεν μπορούν να καλούν διαδικασίες ενός υψηλότερου επιπέδου (βλ. και κεφάλαιο 7).Κατά τη διαδικασία της σύνθεσης (σύνδεσης των επιμέρους ενοτήτων σε ενιαίο πρόγραμμα) ιδιαίτερη προσοχή πρέπει να δοθεί στις μεταβλητές, που ορίζονται και χρησιμοποιούνται σε κάθε ενότητα.

Μεταβλητή *(variable)* ονομάζεται κάθε σύμβολο, χαρακτήρας ή ακολουθία χαρακτήρων, που αναφέρεται σε κάποια τιμή αποθηκευμένη στη μνήμη του υπολογιστή. Κάθε μεταβλητή που χρησιμοποιείται σε ένα πρόγραμμα δεσμεύει μία ή περισσότερες θέσεις μνήμης και μπορεί να αλλάζει τιμή κατά τη διάρκεια της εκτέλεσης του.

Ορισμός

Η μεγάλη σπουδαιότητα που έχουν οι μεταβλητές κατά τη σύνθεση των επιμέρους ενοτήτων της εφαρμογής οφείλεται στο γεγονός ότι οι μεταβλητές πρέπει να ορίζονται και να χρησιμοποιούνται με ομοιόμορφο τρόπο από όλες τις ενότητες. Μερικοί σχεδιαστές λογισμικού δεν επιτρέπουν την επικοινωνία των μεταβλητών μεταξύ των διαδικασιών, ενώ άλλοι σχεδιαστές λογισμικού χρησιμοποιούν μεταβλητές **σφαιρικής εμβέλειας** *(global scope variables)*, που χρησιμοποιούνται σε περισσότερα του ενός κομμάτια του προγράμματος. Οι μεταβλητές σφαιρικής εμβέλειας έχουν την ιδιότητα να είναι προσπελάσιμες από όλα τα μέρη του προγράμματος. Ένας λογικός συμβιβασμός στις παραπάνω θεωρήσεις είναι τα δεδομένα να επικοινωνούν μεταξύ των επιπέδων, μέσω **παραμέτρων** *(parameters)* οι οποίες είναι οι μεταβλη-

τές επικοινωνίας μεταξύ διαφορετικών επιπέδων και να επιτρέπεται η είσοδος **ιδιωτικών μεταβλητών** *(private scope variables)* σε συναρτήσεις ή τμήματα του προγράμματος του ίδιου ιεραρχικού επιπέδου. Οι ιδιωτικές μεταβλητές εξασφαλίζουν την προστασία των αποθηκευμένων ή μεταδιδόμενων δεδομένων σε ένα ιεραρχικό επίπεδο. Αυτή η προσέγγιση γίνεται ιδιαίτερα ελκυστική, όταν κάθε ιεραρχικό επίπεδο του προγράμματος αναγνωρίζεται ως επίπεδο αφαίρεσης του συστήματος. Κάθε επίπεδο αφαίρεσης του συστήματος παρέχει λειτουργίες, που στηρίζουν το επόμενο επίπεδο της ιεραρχίας και υποστηρίζεται από τα επίπεδα που βρίσκονται κάτω από αυτό. Επιπρόσθετα υπάρχουν και οι **τοπικές μεταβλητές** *(local scope variables)* οι οποίες δρουν στο πλαίσιο της διαδικασίας ή συνάρτησης που έχουν δηλωθεί.



Σχήμα 10.2. - Διαγραμματική αναπαράσταση σύνθεσης λογισμικού από κάτω προς τα πάνω.

Κατά τη διάρκεια της σύνθεσης λογισμικού από κάτω προς τα επάνω *(bottom up implementation)* μπορούν να χρησιμοποιηθούν μερικές εικονικές συναρτήσεις *(dummy functions)* με ονόματα σχετικά με τις επιδιωκόμενες λειτουργίες, έτσι ώστε να είναι δυνατός ο έλεγχος με τη δομή του προγράμματος και βαθμιαία να παρέχεται η δυνατότητα της αντικατάστασης των εικονικών συναρτήσεων με λειτουργικές συναρτήσεις.

Ορισμός

Η σύνθεση των απλών προτύπων (μοντέλων) υλοποιήσεων σε ένα πιο σύνθετο και ολοκληρωμένο πρότυπο, που διαγραμματικά απεικονίζεται στο παραπάνω **σχήμα 10.2,**

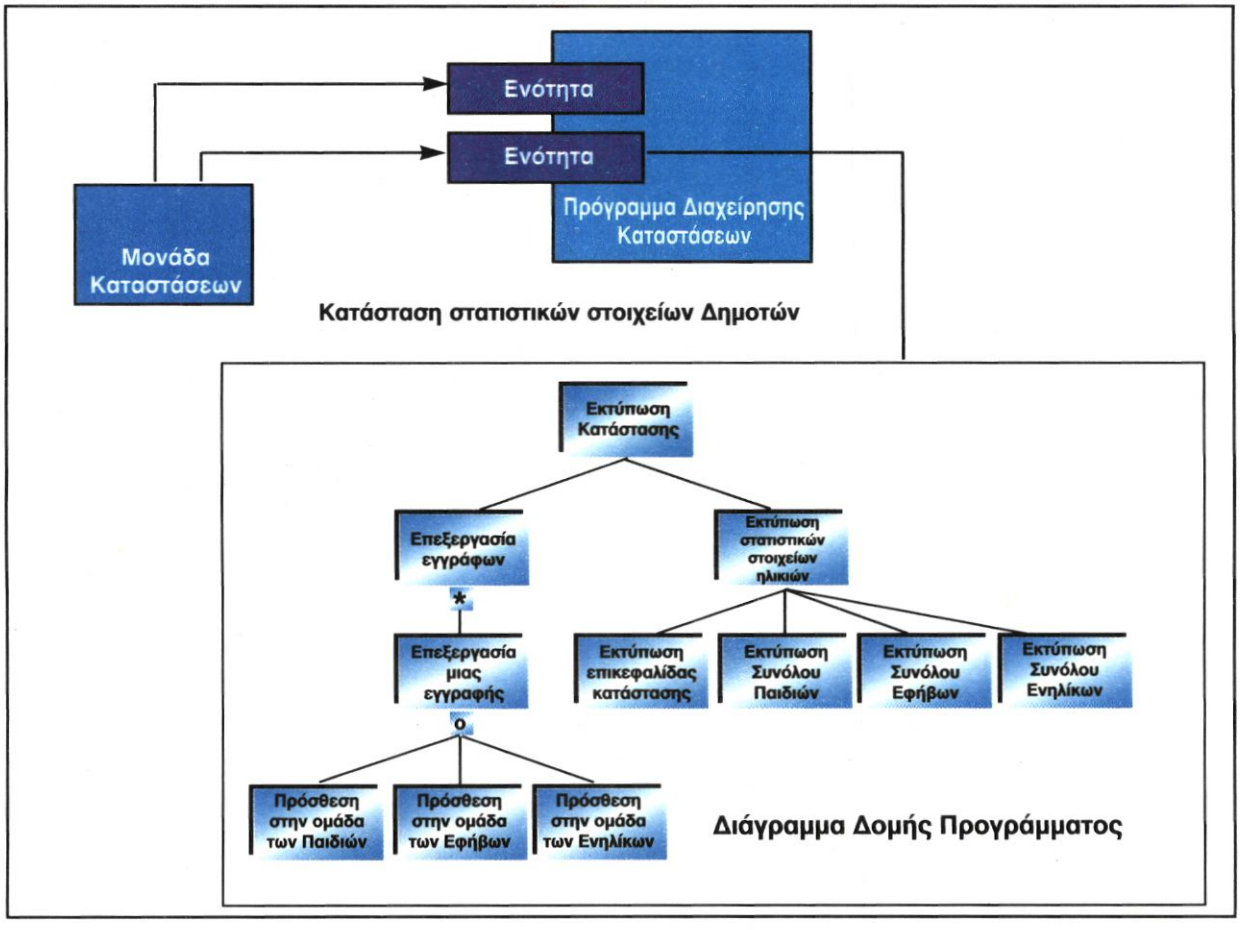
παρέχει μια τακτική και συστηματική οργάνωση για την ανάπτυξη λογισμικού υψηλής ποιότητας και πραγματοποιείται από ομάδες, οι οποίες εφαρμόζουν σε κάθε στάδιο υλοποίησης των προτύπων υλοποιήσεων τις αρχές του προγραμματισμού με αξιοπιστία, που αναφέρθηκαν στο προηγούμενο κεφάλαιο.

Παράδειγμα:

Προκειμένου να γίνει κατανοητή η σύνθεση λογισμικού "από κάτω προς τα πάνω" θα χρησιμοποιήσουμε το παράδειγμα εκτύπωσης κατάστασης στατιστικών στοιχείων των δημοτών ενός Δήμου της χώρας μας, που παρουσιάστηκε στο προηγούμενο κεφάλαιο.

Η διαδικασία "Επεξεργασία εγγραφών" εκτελεί επαναληπτικά (σύμβολο "*") "Επεξεργασία μιας Εγγραφής", κατά την οποία πραγματοποιείται μία από τις επιλογές (σύμβολο "ο") "Πρόσθεση στην ομάδα των παιδιών", "Πρόσθεση στην ομάδα των εφήβων" και "Πρόσθεση στην ομάδα των Ενηλίκων".

Οι διεπαφές *(interfaces)* κατασκευάζονται, προγραμματίζονται και ελέγχονται κατά τη διάρκεια της σταδιακής υλοποίησης.



Σχήμα 10.3. - Σύνθεση απλών προτύπων (μοντέλων) υλοποιήσεων Μονάδας Παραγωγής Καταστάσεων: Η προγραμματιστική ενότητα Εκτύπωση Κατάστασης αποτελεί μέρος μιας άλλης προγραμματιστικής ενότητας, που αφορά στη Μονάδα Καταστάσεων και η οποία Διαχειρίζεται τις Καταστάσεις.

Η διαδικασία "Εκτύπωση στατιστικών στοιχείων ηλικιών" εκτελεί αρχικά την διαδικασία "Εκτύπωση Επικεφαλίδας Κατάστασης", "Εκτύπωση Συνόλου Παιδιών", "Εκτύπωση Συνόλου Εφήβων", και τέλος "Εκτύπωση Συνόλου Ενηλίκων". Η προγραμματιστική ενότητα "Εκτύπωση Κατάστασης" εκτελεί αρχικά την διαδικασία "Επεξεργασία Εγγραφών" και στη συνέχεια τη διαδικασία "Εκτύπωση στατιστικών στοιχείων ηλικιών". Όλα τα παραπάνω απεικονίζονται στο προηγούμενο **σχήμα 10.3.**

Η σύνθεση πραγματοποιείται με υλοποίηση των διαδικασιών-συναρτήσεων, που βρίσκονται στο διάγραμμα δομής προγράμματος από το κάθε επίπεδο προς το ανώτερο επίπεδο και από τα αριστερά προς τα δεξιά κατά βήματα, όπως φαίνεται στον παρακάτω **πίνακα 10.1.**

Βήμα 1:	Υλοποίηση Διαδικασίας "Πρόσθεση στην ομάδα των παιδιών"
Βήμα 2:	Υλοποίηση Διαδικασίας "Πρόσθεση στην ομάδα των εφήβων"
Βήμα 3:	Υλοποίηση Διαδικασία "Πρόσθεση στην ομάδα των ενηλίκων"
Βήμα 4:	Υλοποίηση Διαδικασίας "Επεξεργασία μιας εγγραφής"
	Σύνθεση των διαδικασιών Βημάτων 1 έως 3
Βήμα 5:	Υλοποίηση Διαδικασίας "Εκτύπωση Επικεφαλίδας Κατάστασης"
Βήμα 6:	Υλοποίηση Διαδικασίας "Εκτύπωση Συνόλου Παιδιών"
Βήμα 7:	Υλοποίηση Διαδικασίας "Εκτύπωση Συνόλου Εφήβων"
Βήμα 8:	Υλοποίηση Διαδικασίας "Εκτύπωση Συνόλου Ενηλίκων"
Βήμα 9:	Υλοποίηση Διαδικασίας "Επεξεργασία Εγγραφών" Σύνθεση διαδικασίας Βήματος 4
Βήμα 10:	Υλοποίηση Διαδικασίας "Εκτύπωση στατιστικών στοιχείων ηλικιών" Σύνθεση των διαδικασιών Βημάτων 5 έως 8
Βήμα 11:	Υλοποίηση Διαδικασίας "Εκτύπωση Κατάστασης" Σύνθεση των διαδικασιών Βημάτων 9 έως 10

Πίνακας 10.1. - Ακολουθία βημάτων υλοποίησης των διαδικασιών και της σύνθεσης στην Προγραμματιστική Ενότητα "Εκτύπωση Κατάστασης".

Σημείωση:

Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 10, Μάθημα 1ο, Δραστηριότητα 1.

10.2. ΕΠΑΛΗΘΕΥΣΗ ΚΑΙ ΕΠΙΚΥΡΩΣΗ

Η Επαλήθευση *(Verification)* και η Επικύρωση *(Validation)* ενός συστήματος λογισμικού είναι μία συνεχής διαδικασία σε κάθε στάδιο της διαδικασίας ανάπτυξης λογισμικού.
Η διαδικασία αυτή (V&V) έχει δύο αντικείμενα:

- Τον εντοπισμό των ατελειών του συστήματος.
- Την εξασφάλιση του αν το σύστημα είναι χρήσιμο ή όχι σε οποιαδήποτε λειτουργική περίπτωση.

Η βασική διαφορά ανάμεσα στην Επαλήθευση και την Επικύρωση συνοψίζεται στα εξής (Boehm 1979):
Επαλήθευση: Φτιάχνεται το σωστό προϊόν;
Επικύρωση: Το προϊόν φτιάχνεται σωστά;

Η **Επαλήθευση** και η **Επικύρωση** (V&V) εξασφαλίζει ότι το λογισμικό ικανοποιεί τις απαιτήσεις του πελάτη και ότι οι απαιτήσεις καλύπτουν τις προδιαγραφές ανάπτυξης του παραγωγού του λογισμικού.

Για την ικανοποίηση των στόχων της διαδικασίας V&V υπάρχουν δύο είδη ελέγχου:
- **Στατιστικός έλεγχος** *(statistical testing)*: Αποβλέπει στο σχεδιασμό ελέγχων εισόδων/εξόδων για την εκτίμηση της λειτουργικής αξιοπιστίας του συστήματος.
- **Έλεγχος ατελειών** *(defect testing)*. Αποβλέπει στο σχεδιασμό ελέγχων για την αποκάλυψη ατελειών στο σύστημα, οι οποίες μπορούν να εντοπιστούν και να διορθωθούν με την βοήθεια της εκσφαλμάτωσης *(debugging)*.

10.3. ΕΛΕΓΧΟΙ

10.3.1. Διαδικασία Ελέγχου

Όπως έχει αναφερθεί σε προηγούμενα κεφάλαια, τα μεγάλα συστήματα αποτελούνται από υποσυστήματα *(sub-systems)*, τα οποία αποτελούνται από μονάδες *(modules)*, οι οποίες αποτελούνται από ενότητες που περιέχουν **διαδικασίες** *(procedures)* και **συναρτήσεις** *(functions)*. Εάν προκύψει ένα σφάλμα σε κάποιο από τα απλούστερα στοιχεία που βρίσκονται στα χαμηλότερα ιεραρχικά επίπεδα, τότε κατά τη σύνθεση το σφάλμα μεταφέρεται στα πιο σύνθετα τμήματα των ανώτερων ιεραρχικών επιπέδων του συστήματος. Π.χ. αν οριστεί λάθος μια συνάρτηση ή μια διαδικασία, τότε το λάθος αυτό μεταφέρεται και σε όσες μονάδες τη χρησιμοποιούν, καθώς επίσης και σε όσα υποσυστήματα περιλαμβάνουν τις παραπάνω μονάδες. Επομένως, η διαδικασία ελέγχου θα πρέπει να γίνεται σε στάδια σε συνδυασμό με την πρόοδο στην υλοποίηση του συστήματος.

Σχήμα 10.4. - Διαγραμματική Αναπαράσταση Διαδικασίας Ελέγχου

Η διαδικασία ελέγχου του λογισμικού που διαγραμματικά παριστάνεται στο **σχήμα 10.4**, αποτελείται από έξι στάδια:

- **Έλεγχος μονάδων** *(unit testing)*: Οι μονάδες ελέγχονται για να εξακριβωθεί ότι λειτουργούν

σωστά. Κάθε ενότητα αντιμετωπίζεται σαν μία ανεξάρτητη οντότητα, που δεν χρειάζεται άλλες συνιστώσες προκειμένου να ελεγχθεί.

- **Έλεγχος ενοτήτων** *(module testing):* Η φάση αυτή περιλαμβάνει τον ανεξάρτητο έλεγχο του συνόλου των ενοτήτων που υπάρχουν στα υποσυστήματα. Τα υποσυστήματα μπορεί να έχουν σχεδιαστεί και υλοποιηθεί ανεξάρτητα και τα πιο συνήθη προβλήματα στα μεγάλα συστήματα λογισμικού προκύπτουν στο στάδιο αυτό και αφορούν στον τρόπο διεπαφής των μονάδων των υποσυστημάτων.
- **Έλεγχος υποσυστημάτων** *(subsystem testing):* Οι προγραμματιστικές ενότητες που συνιστούν ένα υποσύστημα ελέγχονται σε συνεργασία μεταξύ τους, προκειμένου να ελεγχθεί η ανεξάρτητη λειτουργία των υποσυστημάτων.
- **Έλεγχος συστήματος** *(system testing):* Τα υποσυστήματα ολοκληρώνονται για να σχηματίσουν το σύστημα. Η διαδικασία ελέγχου επικεντρώνεται στον εντοπισμό λαθών, που προκύπτουν από την επικοινωνία των υποσυστημάτων και των συνιστωσών τους.
- **Έλεγχος ολοκλήρωσης** *(integration testing)* Σε μεγάλα έργα ανάπτυξης λογισμικού αυτός ο τύπος ελέγχου πραγματοποιείται από έναν οργανισμό ελέγχου του λογισμικού. Σε μικρά συστήματα ο έλεγχος ολοκλήρωσης πραγματοποιείται κάθε φορά που ενότητες συνδυάζονται για να δημιουργήσουν μία μονάδα, οποτεδήποτε μονάδες συνδυάζονται για να δημιουργήσουν υλοποιήσεις υποσυστημάτων, καθώς και κάθε φορά που υποσυστήματα συνδυάζονται για να δημιουργήσουν το σύστημα και αφορά στον τρόπο επικοινωνίας των συνιστωσών λογισμικού.
- **Έλεγχος αποδοχής** *(acceptance testing):* Είναι το τελευταίο στάδιο της διαδικασίας ελέγχου πριν το σύστημα γίνει αποδεκτό για χρήση. Αναμιγνύει έλεγχο του συστήματος με πραγματικά δεδομένα αντί των τεχνητών δεδομένων (π.χ. δεδομένων προσομοίωσης), που ενδεχομένως είχαν χρησιμοποιηθεί κατά την φάση ανάπτυξης του συστήματος. Συνήθως εδώ αποκαλύπτονται λάθη και παραλείψεις στον καθορισμό απαιτήσεων του συστήματος. Επίσης, οι απαιτήσεις ίσως δεν αντικατοπτρίζουν τις ευκολίες και την απόδοση που επιζητεί ο χρήστης και ο έλεγχος αυτός ίσως αποκαλύψει μειωμένη απόδοση και λειτουργικότητα.

10.3.2. Σχεδιασμός του Ελέγχου

Ο έλεγχος του συστήματος έχει υψηλό κόστος. Επομένως προσεκτικός σχεδιασμός είναι αναγκαίος για να ελεγχθούν τα κόστη αυτής της διαδικασίας.

Οι εργασίες (συνιστώσες), που περιλαμβάνει μια τυπική διαδικασία ελέγχου του λογισμικού περιγράφονται στον παρακάτω **πίνακα 10.2.**

• ΔΙΑΔΙΚΑΣΙΑ ΕΛΕΓΧΟΥ : Περιγραφή των φάσεων ελέγχου.
• ΕΛΕΓΧΟΣ ΑΠΑΙΤΗΣΕΩΝ: Οι χρήστες ενδιαφέρονται εάν το σύστημα ικανοποιεί τις απαιτήσεις τους και ο έλεγχος πρέπει να σχεδιαστεί έτσι ώστε όλες οι απαιτήσεις να ελεγχθούν ξεχωριστά.
• ΕΛΕΓΧΟΜΕΝΑ ΣΤΟΙΧΕΙΑ: Πρέπει να καθοριστούν τα προϊόντα λογισμικού που θα ελεγχθούν.
• ΠΡΟΓΡΑΜΜΑ ΕΛΕΓΧΟΥ: Πρέπει να φτιαχτεί χρονοδιάγραμμα του ελέγχου.
• ΔΙΑΔΙΚΑΣΙΕΣ ΕΛΕΓΧΟΥ: Δεν είναι εύκολη η εκτέλεση των ελέγχων. Τα αποτελέσματα τους πρέπει να καταγραφούν συστηματικά και να επανελεγχθούν σε όλη τη φάση ελέγχου, για να ελεγχθεί εάν εκτελέστηκαν σωστά.
• ΑΠΑΙΤΗΣΕΙΣ ΣΕ ΥΛΙΚΟ ΚΑΙ ΛΟΓΙΣΜΙΚΟ: Καθορισμός εργαλείων λογισμικού και υποδομής σε υλικό, που απαιτούνται για να γίνει ο έλεγχος.
• ΠΕΡΙΟΡΙΣΜΟΙ: Όλοι οι περιορισμοί που αφορούν στη διαδικασία ελέγχου πρέπει να καταγραφούν (π.χ. έλλειψη προσωπικού).

Πίνακας 10.2. - Βασικές συνιστώσες Σχεδίου Ελέγχου.

10.3.3. Προδιαγραφές του Ελέγχου

Ο έλεγχος δε γίνεται μόνο για τον παραγωγό του λογισμικού, ώστε να μπορέσει να εντοπίσει τα σφάλματα και τις ατέλειες του παραγόμενου προϊόντος, αλλά και για τον πελάτη, ο οποίος ενδιαφέρεται να βεβαιωθεί ότι η εφαρμογή που παραλαμβάνει έχει περάσει με επιτυχία όλα τα απαραίτητα τεστ ελέγχου καλής λειτουργίας. Για το λόγο αυτό η διαδικασία ελέγχου πρέπει να τεκμηριώνεται με πληρότητα και σαφήνεια.

Οι **προδιαγραφές** του ελέγχου περιγράφουν σε φυσική γλώσσα και με συντομία τη φιλοσοφία του ελέγχου, τη χρονική στιγμή εκτέλεσης του ελέγχου και τη σκοπιμότητα του ελέγχου.

Η **τεκμηρίωση** του ελέγχου περιλαμβάνει τα στοιχεία που περιγράφονται στον παρακάτω **πίνακα 10.3.**

• ΚΑΤΑΣΤΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΤΟΥ ΕΛΕΓΧΟΥ: Ποιοι έλεγχοι πραγματοποιήθηκαν και τι αποτελέσματα έδωσαν.
• ΚΑΤΑΣΤΑΣΗ ΣΦΑΛΜΑΤΩΝ: Τι σφάλματα παρουσιάστηκαν κατά τη διαδικασία του ελέγχου, γιατί προέκυψαν, πόσο σημαντικά είναι και πώς αντιμετωπίζονται.
• ΕΙΔΙΚΕΣ ΠΕΡΙΠΤΩΣΕΙΣ: Υπάρχουν κάποιες ειδικές περιπτώσεις, που είναι δυνατό να παρουσιαστεί πρόβλημα κατά τη λειτουργία της εφαρμογής ή που η εφαρμογή δεν καλύπτει πλήρως τις απαιτήσεις του πελάτη;
• ΣΥΣΤΑΣΕΙΣ: Παρατηρήσεις και σχόλια που αφορούν στη χρήση, την ασφάλεια, τη συντήρηση και τη διαχείριση της εφαρμογής.

Πίνακας 10.3. - Τεκμηρίωση του ελέγχου.

10.3.4. Στρατηγικές Ελέγχου

Κάθε στρατηγική έχει τα πλεονεκτήματα και τα μειονεκτήματα της. Στα μεγάλα συστήματα χρησιμοποιείται μίγμα των στρατηγικών. Η τεχνική που θα υιοθετηθεί εξαρτάται από την οργάνωση, την εφαρμογή και τους τεχνολόγους λογισμικού που συμμετέχουν στο έργο λογισμικού. Οι τεχνικές που παρουσιάζονται στη συνέχεια εντάσσονται στο στάδιο του ελέγχου ολοκλήρωσης.

Έλεγχος από πάνω προς τα κάτω *(top down testing)*. Ο έλεγχος γίνεται σε επίπεδο κάθε υποσυστήματος. Στο επίπεδο αυτό οι μονάδες που περιέχει κάθε υποσύστημα αναφέρονται, αλλά δε συμμετέχουν στον έλεγχο (δηλ. δεν ελέγχονται). Μετά τον έλεγχο των υποσυστημάτων γίνεται έλεγχος των μονάδων με τον ίδιο τρόπο, ξεκινώντας από τις υψηλότερες στην ιεραρχία μονάδες *(modules)* και σύμφωνα με το ιεραρχικό διάγραμμα του λογισμικού. Στο τέλος γίνεται τοποθέτηση των ενοτήτων προγράμματος στον κώδικα και αυτός ελέγχεται.

Έλεγχος από κάτω προς τα πάνω *(bottom up testing)*.Ο έλεγχος ξεκινά από τις ενότητες που βρίσκονται χαμηλότερα στην ιεραρχία, αμέσως μόλις αυτές υλοποιούνται. Ο έλεγχος στη συνέχεια εκτελείται στα ανώτερα ιεραρχικά επίπεδα, καθώς προχωρεί η σύνθεσή τους από κάτω προς τα πάνω σε μονάδες, υποσυστήματα και συστήματα.

Σημείωση:

Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 10, Μάθημα 2ο, Δραστηριότητα 2.

Ανακεφαλαίωση / περίληψη

Η ανάπτυξη λογισμικού κατά την σύνθεση των επιμέρους εργασιών "από κάτω προς τα πάνω" αρχίζει με την σύνταξη των απλούστερων διαδικασιών, συναρτήσεων και ενοτήτων του προγράμματος, οι οποίες χρησιμοποιούνται στη συνέχεια για την δόμηση πιο σύνθετων τμημάτων. Η διαδικασία επαναλαμβάνεται μέχρι την πλήρη ανάπτυξη του προγράμματος.

Οι έλεγχοι του λογισμικού αποφασίζονται κατά τη διάρκεια της ανάλυσης των απαιτήσεων και τροποποιούνται κατά τη διάρκεια της φάσης σχεδιασμού των ελέγχων. Κατά τον καθορισμό των προδιαγραφών του ελέγχου είναι αναγκαία η ύπαρξη δοκιμαστικών δεδομένων, αναμενόμενων αποτελεσμάτων, κριτηρίων εισόδου και κριτηρίων εξόδου. Μετά την ολοκλήρωση του ελέγχου είναι αναγκαία η σύνταξη αναφοράς αποτελεσμάτων του ελέγχου.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Να υλοποιήσετε το σύστημα μενού της Εφαρμογής Βιβλιοθήκης χρησιμοποιώντας σύνθεση "από κάτω προς τα πάνω"
2. Να υλοποιήσετε σύνθεση της μονάδας των πράξεων Διαχείρισης Βιβλίων της Εφαρμογής Βιβλιοθήκης με το Σύστημα μενού της Εφαρμογής Βιβλιοθήκης.
3. Επιλέξτε και εφαρμόστε στρατηγική ελέγχου για το λογισμικό που αναπτύξατε για την Εφαρμογή Βιβλιοθήκης και τεκμηριώστε κατάλληλα τον έλεγχο.
4. Διατυπώστε την ακολουθία βημάτων υλοποίησης των διαδικασιών και της σύνθεσης στην Προγραμματιστική Ενότητα αναφοράς (κατάστασης) με τους συνδρομητές της Βιβλιοθήκης που έχουν δανειστεί ένα βιβλίο με συγκεκριμένο κωδικό ταξινομημένους ως προς το ονοματεπώνυμο. Η κατάσταση να έχει τη μορφή:

	ΚΩΔΙΚΟΣ ΒΙΒΛΙΟΥ : ########	
ΕΠΩΝΥΜΟ	ΟΝΟΜΑ	ΗΜΕΡΟΜΗΝΙΑ ΕΠΙΣΤΡΟΦΗΣ

Γλωσσάριο - Ευρετήριο όρων

επαλήθευση και επικύρωση *(verification & validation)* - Γενικός όρος για δοκιμές που πιστοποιούν ότι οι απαιτήσεις και οι στόχοι ενός συστήματος πληρούνται. Η επαλήθευση αξιολογεί με ποσοτικό τρόπο πόσο καλά οι απαιτήσεις ικανοποιούν τους στόχους του συστήματος [208]

.

έλεγχος λογισμικού *(software testing)* - Η διαδικασία της εκτέλεσης ενός προγράμματος με δοκιμαστικά δεδομένα και σύγκριση των αποτελεσμάτων του με αποτελέσματα υπολογισμών, που έγιναν με το χέρι ή με τα αποτελέσματα μιας άλλης δοκιμασμένης μεθόδου [209].

έλεγχος ενοτήτων *(module testing)* - Η φάση αυτή περιλαμβάνει τον ανεξάρτητο έλεγχο του συνόλου των ενοτήτων που υπάρχουν στα υποσυστήματα. Τα υποσυστήματα μπορεί να έχουν σχεδιαστεί και υλοποιηθεί ανεξάρτητα και τα πιο συνήθη προβλήματα στα μεγάλα συστήματα λογισμικού προκύπτουν στο στάδιο αυτό και αφορούν στον τρόπο διεπαφής των μονάδων των υποσυστημάτων [210].

έλεγχος υποσυστημάτων *(subsystem testing)* - Οι προγραμματιστικές ενότητες που συνιστούν ένα υποσύστημα ελέγχονται σε συνεργασία μεταξύ τους, προκειμένου να ελεγχθεί η ανεξάρτητη λειτουργία των υποσυστημάτων [210].

ολοκληρωμένη σύνθεση "από κάτω προς τα πάνω" *(integrated bottom up development)* - Μέθοδος σχεδίασης των συνιστωσών του λογισμικού κατά την οποία πρώτα ορίζονται τα χαμηλότερα επίπεδα, στη συνέχεια τα αμέσως ανώτερα από αυτά κ.ο.κ. μέχρι τον ορισμό του ανώτατου ιεραρχικού επιπέδου, την πλήρη δηλαδή περιγραφή της εφαρμογής λογισμικού [205].

υλοποίηση (λογισμικού) "από κάτω προς τα πάνω" *(bottom up implementation)* - Μέθοδος υλοποίησης του έργου ανάπτυξης του λογισμικού μέσω προγραμματισμού "από το ειδικό στο γενικό" [206].

Βιβλιογραφία

Sommerville, "Software Engineering", Addison-Wesley, 1985.

Stevens W., "Using Structured Design: How to Make Programs Simple", Changeable, Flexible and Reusable, Wiley-Interscience, 1981.

DeMarco T., "Structured Analysis and System Specification", Yourdon Press, 1978.

Γαρίδης Π., Δεληγιαννάκης Μ., "Σύγχρονο λεξικό Πληροφορικής", Εκδόσεις Δίαυλος, 1989.

ΚΕΦΑΛΑΙΟ 11

ΔΟΚΙΜΗ ΜΕ ΠΡΑΓΜΑΤΙΚΑ ΔΕΔΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ

11.1. Δοκιμαστικά και πραγματικά δεδομένα
11.2. Δοκιμές οριακών περιπτώσεων δεδομένων

ΕΙΣΑΓΩΓΗ

Όσο προσεκτικά και αν έχει γίνει η σχεδίαση και η υλοποίηση μιας εφαρμογής λογισμικού, είναι σχεδόν αδύνατο να μη περιέχει λάθη. Τα σφάλματα αυτά δεν αφορούν συντακτικά ή ορθογραφικά λάθη κατά την υλοποίηση του λογισμικού αλλά αφορούν σφάλματα λειτουργίας. Για το λόγο αυτό είναι απαραίτητο η εφαρμογή να υποβάλλεται σε έλεγχο, πριν παραδοθεί στον τελικό χρήστη. Η διαδικασία των ελέγχων σε γενικές γραμμές είναι η εξής: Δοκιμάζεται η λειτουργία της εφαρμογής κάτω από διαφορετικές συνθήκες και συνδυασμούς περιπτώσεων και συγκρίνεται η συμπεριφορά της με την αναμενόμενη. Στις περιπτώσεις που διαπιστώνεται διαφορά υπάρχει σφάλμα.

- Πώς, όμως, ανακαλύπτονται τα σφάλματα;
- Τι είδους σφάλματα είναι δυνατό να προκύψουν;

Οι έλεγχοι με δοκιμαστικά δεδομένα, οι έλεγχοι οριακών τιμών, οι έλεγχοι τύπου "black box" και "white box" και οι έλεγχοι οριακών περιπτώσεων είναι μερικές από τις πιο συνηθισμένες τεχνικές και στρατηγικές, που εφαρμόζονται για την αποκάλυψη των σφαλμάτων, που μπορεί να περιέχουν οι εφαρμογές λογισμικού.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να εκτιμήσει την αποτελεσματικότητα της λύσης ανάπτυξης λογισμικού.
2. Να μπορεί να εφαρμόζει το πρότυπο λογισμικού που δημιούργησε σε πραγματικά δεδομένα.
3. Να μπορεί να εντοπίζει περιπτώσεις δεδομένων, που παρουσιάζουν προβλήματα στη λύση.

11.1. ΔΟΚΙΜΑΣΤΙΚΑ ΚΑΙ ΠΡΑΓΜΑΤΙΚΑ ΔΕΔΟΜΕΝΑ

Από τη διαδικασία ελέγχου, που περιγράφηκε στο προηγούμενο κεφάλαιο, είναι πολύ φυσικό να προκύψουν σφάλματα στη λειτουργία της εφαρμογής. Έτσι, ενώ ο έλεγχος *(testing)* στοχεύει στην αποκάλυψη των σφαλμάτων του λογισμικού, η **εκσφαλμάτωση** *(debugging)* αποσκοπεί στην εξουδετέρωση αυτών των σφαλμάτων.

Ο εντοπισμός σφαλμάτων με χρήση δοκιμαστικών δεδομένων προϋποθέτει να γνωρίζουμε εκ των προτέρων ποια θα πρέπει να είναι σε κάθε περίπτωση η συμπεριφορά της εφαρμογής, δηλ. ποια πρέπει να είναι τα εξερχόμενα δεδομένα, σύμφωνα με τις προδιαγραφές και τις απαιτήσεις ανάπτυξης της εφαρμογής.

Οι έλεγχοι πραγματοποιούνται από ομάδες προγραμματιστών, οι οποίοι δε μετείχαν στο έργο ανάπτυξης του λογισμικού και γίνονται σύμφωνα με το σχέδιο ελέγχου του λογισμικού. Υπάρχουν διάφοροι τύποι σφαλμάτων *(error types)* του λογισμικού. Οι κυριότεροι από αυτούς παρουσιάζονται στον παρακάτω **πίνακα 11.1**.

Αλγοριθμικά *(algorithmic)* τα οποία οφείλονται στην εκτέλεση εσφαλμένων ελέγχων (Βλέπε 11.1.2. σφάλματα σύγκρισης και σφάλματα ελέγχου ροής).
Υπολογιστικά *(computational)* ή ακριβείας *(precision)* όπως είναι, για παράδειγμα, εντοπισμός μη σωστού αποτελέσματος.
Αποσυντονισμού *(stress)* ή υπερφόρτωσης *(overload)* όπως είναι, για παράδειγμα, η περίπτωση που η μνήμη δεν επαρκεί.
Εξάντλησης χωρητικότητας στατικών δομών δεδομένων *(capacity of boundary)*, όπως είναι, για παράδειγμα, η χρήση κενών πινάκων ή η προσπάθεια ενημέρωσης ενός πίνακα που είναι πλήρης.
Χρονικού Συγχρονισμού *(timing)* και συνεργασίας *(coordination)* όπως είναι, για παράδειγμα, η ταυτόχρονη χρήση δεδομένων.
Απόδοσης *(performance)* όπως είναι, για παράδειγμα, η περίπτωση της πολύ αργής εκτέλεσης του προγράμματος.
Ανάκτησης *(recovery)*, όπως είναι, για παράδειγμα, ο χειρισμός της περίπτωσης καταστροφής του σκληρού δίσκου.
Χειρισμού λαθών στο υλικό ή το λογισμικό *(hardware* ή *system errors)* όταν δεν προβλέπονται ή δε λειτουργούν αποτελεσματικά οι προβλεπόμενες διαδικασίες λόγω βλαβών στο υλικό ή το λογισμικό.

Πίνακας 11.1. - Τύποι σφαλμάτων του λογισμικού.

Η ανίχνευση όλων των παραπάνω τύπων σφαλμάτων είναι ουσιαστική, τόσο κατά τον **έλεγχο των μονάδων** *(module testing)*, όπου οι λειτουργίες των μονάδων ελέγχονται ανεξάρτητα, όσο και κατά τον **έλεγχο ολοκλήρωσης** *(integration testing)*, όπου οι συνιστώσες του συστήματος ελέγχονται για τον τρόπο λειτουργίας τους σύμφωνα με τη σύνθεση της σχεδίασης τους.

Ορισμός

Τα **δοκιμαστικά δεδομένα** *(test data)* είναι δεδομένα που κατασκευάζονται από την ομάδα ελέγχου του λογισμικού, με σκοπό να εξακριβωθεί εάν έχουν υλοποιηθεί σωστά καθορισμένες λειτουργίες του συστήματος.

Για τα δοκιμαστικά δεδομένα είναι αναγκαίο να υπολογιστούν πριν την εκτέλεση του προγράμ-

ματος, με εφαρμογή των αλγορίθμων του λεπτομερούς σχεδιασμού, τα **αναμενόμενα αποτελέσματα** *(expected results)*, με βάση τα οποία είναι δυνατό να ελεγχθεί εάν η υλοποίηση των λειτουργιών έχει πραγματοποιηθεί σωστά. Τα πραγματικά δεδομένα και τα αντίστοιχα πραγματικά αναμενόμενα αποτελέσματα μπορούν επίσης να χρησιμοποιηθούν κατά των έλεγχο σφαλμάτων και συνήθως είναι διαθέσιμα από στοιχεία εντύπων, τα οποία τηρούνται στο υπάρχον χειρογραφικό σύστημα ή παράγονται από το υπάρχον χειρογραφικό σύστημα.

Με τους ελέγχους τύπου "black box", μόνο συμπτωματικά μπορεί να αποκαλυφθούν σφάλματα ως προς την ορθή λειτουργία κάποιων σπάνια εκτελούμενων εντολών, η ακολουθούμενων διακλαδώσεων του προγράμματος. Αντίθετα, οι έλεγχοι τύπου "white box" σχεδιάζονται ακριβώς για να ελεγχθεί η ορθότητα αυτών των εντολών και διακλαδώσεων του προγράμματος.

11.1.1. Μέθοδοι δημιουργίας δοκιμαστικών δεδομένων

Υπάρχουν δύο είδη ελέγχων με δοκιμαστικά δεδομένα: Οι έλεγχοι τύπου "black box" και οι έλεγχοι τύπου "white box".

- **Οι έλεγχοι τύπου "Black Box"** *(black box testing)* σχεδιάζονται χωρίς να έχει ο σχεδιαστής του ελέγχου προγενέστερη γνώση για την σχεδίαση του προγράμματος ή την υλοποίηση του. Οι έλεγχοι αυτοί στηρίζονται στις προδιαγραφές των διεπαφών του λογισμικού και για να πραγματοποιηθούν, απαιτούν την ύπαρξη δοκιμαστικών δεδομένων και αναμενόμενων αποτελεσμάτων.

- **Οι έλεγχοι τύπου "White Box"** *(white box testing)* σχεδιάζονται με γνώση το διάγραμμα δομής προγράμματος ή την υλοποίηση του. Οι έλεγχοι αυτοί αποσκοπούν στο να αναδείξουν λογικά προγραμματιστικά σφάλματα και απαιτούν να υπάρχουν δοκιμαστικά δεδομένα, ώστε να ελεγχθεί ως προς την ορθότητα λειτουργίας της κάθε **εντολή** *(statement)* και κάθε **διακλάδωση** *(branch,path)* του προγράμματος. Αυτά τα λογικά σφάλματα ονομάζονται **σφάλματα προγραμματισμού** *(program errors)* και γίνονται από τον προγραμματιστή κατά τη διάρκεια σύνταξης του προγράμματος.

Προκειμένου να δημιουργηθούν δοκιμαστικά δεδομένα κατάλληλα για την εκτέλεση των παραπάνω τύπων ελέγχων είναι αναγκαίο να ακολουθήσετε τα βήματα του παρακάτω πίνακα 11.2.

- Διαιρέστε τον χώρο δεδομένων σας σε συλλογές διακεκριμένων συνόλων δεδομένων και αντιστοιχείστε σε κάθε συλλογή ένα στοιχείο δεδομένων ως αντιπρόσωπο της συλλογής
- Χρησιμοποιείστε τις περιπτώσεις αντιπροσωπευτικών δεδομένων για να σχεδιάσετε περιπτώσεις ελέγχου
- Καθορίστε ειδικές περιπτώσεις για να ελέγξετε τις **οριακές περιπτώσεις μεταξύ των συλλογών των δεδομένων** *(partition boundaries)*

Πίνακας 11.2. - Στρατηγική δημιουργίας Δοκιμαστικών Δεδομένων

Σημείωση:

Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 11, Μάθημα 1ο, Δραστηριότητες 1, 2, 3.

11.1.2. Στρατηγική εποπτείας σφαλμάτων

Ένα σημαντικό τμήμα της δραστηριότητας εποπτείας σφαλμάτων είναι η χρησιμοποίηση μιας τυπικής κατάταξης των σφαλμάτων, που είναι συνήθη στα προϊόντα λογισμικού, έτσι ώστε να υπάρχει ανεξαρτησία από τον φορέα υλοποίησης. Οι κυριότερες κατηγορίες στις οποίες κατατάσσονται τα συνήθη σφάλματα είναι οι εξής:

- **Σφάλματα αναφοράς σε τιμή μεταβλητής** *(data reference errors).* Προκύπτουν όταν γίνεται προσπάθεια να καταχωρηθεί τιμή εκτός ορίων σε **μια στατική μεταβλητή** *(static variable)* ή όταν εξαντληθεί ο χώρος αποθήκευσης των **δυναμικά κατανεμημένων μεταβλητών** *(dynamic allocated variables).* Οι στατικές μεταβλητές προκαταλαμβάνουν χώρο στην κύρια μνήμη με τη δήλωση της και σύμφωνα με τον τύπο της ενώ οι δυναμικές μεταβλητές δεσμεύουν και αποδεσμεύουν χώρο της κύριας μνήμης, όποτε το θελήσει ο προγραμματιστής. Βλέπε το παράδειγμα που παρουσιάζεται στην ενότητα 11.2. στο οποίο περιγράφεται ο τρόπος ελέγχου καταχώρησης τιμής εκτός ορίων σε μια μεταβλητή τύπου πίνακα.

- **Σφάλματα δηλώσεων μεταβλητών** *(data declaration errors).* Προκύπτουν όταν χρησιμοποιείται μεταβλητή η οποία δεν έχει δηλωθεί, ή όταν γίνεται προσπάθεια ανάθεσης τιμής σε μεταβλητή, η οποία είναι εκτός του πεδίου τιμών του τύπου της. Ένα τυπικό παράδειγμα αυτής της κατηγορίας είναι να αναμένεται από τον χρήστη να πληκτρολογήσει αριθμητική τιμή (πραγματικό αριθμό) σε ένα στοιχείο δεδομένων και αντί αυτού ο χρήστης να πληκτρολογήσει μια ακολουθία χαρακτήρων.

- **Υπολογιστικά σφάλματα** *(computation errors).* Προκύπτουν όταν μια αριθμητική λειτουργία δημιουργεί λάθος. Π.χ., κατά τη διατύπωση πραγματοποίησης μιας υπολογιστικής λειτουργίας στην οποία πραγματοποιείται διαίρεση δεν προηγείται έλεγχος της τιμής του παρανομαστή με αποτέλεσμα να παρουσιαστεί υπολογιστικό σφάλμα όταν αυτός πάρει τιμή μηδέν.

- **Σφάλματα σύγκρισης** *(comparison errors).* Παρουσιάζονται όταν μια δομή ελέγχου δε διατυπωθεί σωστά. Π.χ., χρήση εσφαλμένου λογικού τελεστή κατά την διατύπωση μιας σύνθετης συνθήκης σύγκρισης μέσω της οποίας ο προγραμματιστής επιθυμεί να διατυπώσει το ερωταπάντημα εμφάνισης των υπαλλήλων με μισθούς μεγαλύτερους των 200.000 δρχ. και ταυτόχρονα μικρότερων των 50.000 δραχμών.

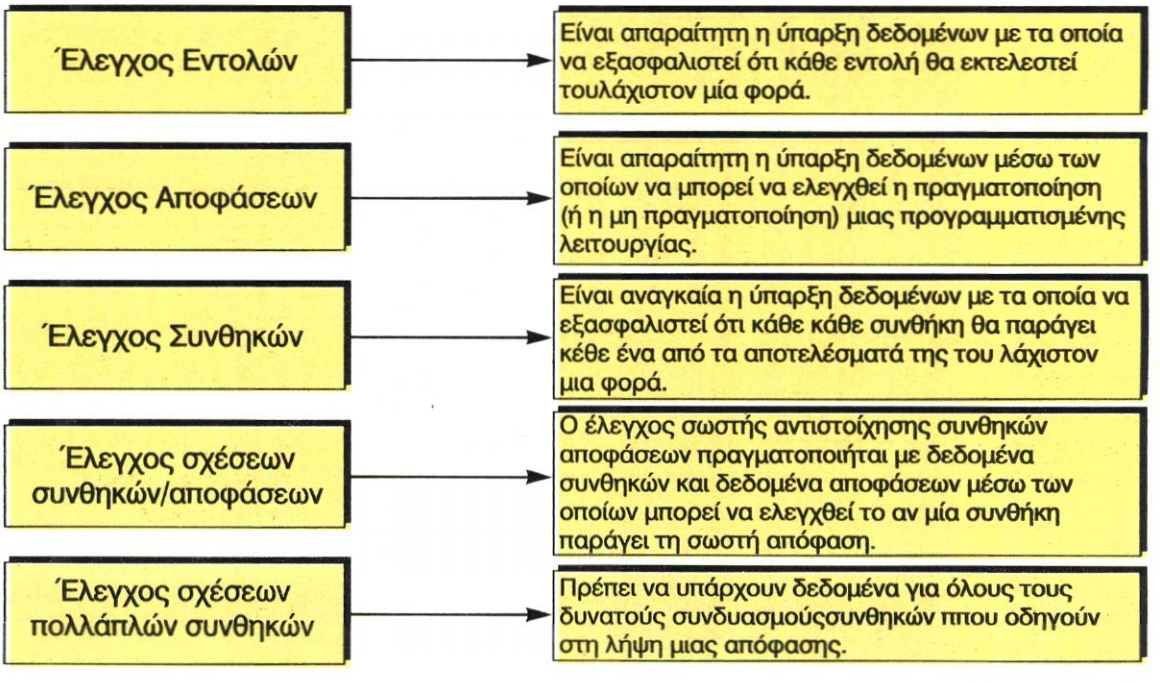
- **Σφάλματα ελέγχου ροής** *(control flow errors).* Παρουσιάζονται όταν η ακολουθία των αλγοριθμικών βημάτων δεν έχει διατυπωθεί σωστά. Π.χ., εσφαλμένη τοποθέτηση της εντολής εκτέλεσης μιας υπολογιστικής λειτουργίας μέσω της οποίας χορηγείται αύξηση 30% επί του βασικού μισθού σε όλους τους υπαλλήλους με μισθό μικρότερο των 50.000 δρχ. με αποτέλεσμα να χορηγηθεί η αύξηση σε όλους τους υπαλλήλους.

- **Σφάλματα διεπαφών** *(interface errors).* Προκύπτουν κατά τη σύνδεση προγραμμάτων με διαφορετικά χαρακτηριστικά. Π.χ., επιθυμούμε να ακούγεται ένα μουσικό κομμάτι με την επιλογή εξόδου από την εφαρμογή λογισμικού το οποίο είναι σε μορφή κατάλληλη να εκτε-

λεστεί από το τυπικό εργαλείο εκτέλεσης εφαρμογών πολυμέσων του υπολογιστικού μας περιβάλλοντος και αυτό δεν εκτελείται διότι το εργαλείο εκτέλεσης πολυμέσων έχει απεγκατασταθεί.

- **Σφάλματα εισόδων/εξόδων** *(input/output errors)*. Προκύπτουν κατά την σύνδεση του προγράμματος με αρχεία ή περιφερειακές συσκευές. Π.χ., ζητάμε εκτύπωση στον εκτυπωτής και αυτός δεν είναι συνδεδεμένος. Λογικά σφάλματα

Η στρατηγική εποπτείας σφαλμάτων που εφαρμόζεται για τον εντοπισμό των **λογικών σφαλμάτων** *(condition/decision testing)* κατά την εφαρμογή των ελέγχων τύπου White Box στηρίζεται στην δημιουργία δεδομένων για να ελέγξετε **εντολές** *(statements)*, προγραμματιστικές **αποφάσεις** *(decisions)*, **συνθήκες** *(conditions)*, **σχέσεις συνθηκών/αποφάσεων** *(condition/decision coverage)* και **σχέσεις πολλαπλών συνθηκών** *(multiple condition coverage)*. Ένα πλάνο εντοπισμού λογικών σφαλμάτων παρουσιάζεται στο **σχήμα 11.1.**



Σχήμα 11.1. - Πλάνο εντοπισμού λογικών σφαλμάτων.

11.2. ΔΟΚΙΜΕΣ ΟΡΙΑΚΩΝ ΠΕΡΙΠΤΩΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Οι περιπτώσεις ελέγχου **οριακών περιπτώσεων *δεδομένων*** *(boundary conditions)* είναι πάρα πολύ σημαντικές γιατί αποκαλύπτουν τη συμπεριφορά του προγράμματος, όταν διοχετευτούν σε αυτό δεδομένα εισόδου, τα οποία βρίσκονται στα όρια του διαστήματος του πεδίου ορισμού των δεδομένων εισόδου. Η **μέθοδος δοκιμών με οριακές περιπτώσεις δεδομένων** *(boundary value analysis)* αποτελεί μια από τις πλέον χρήσιμες μεθόδους σχεδίασης περιπτώσεων για έλεγχο.

Η αναγκαιότητα δοκιμής με οριακές περιπτώσεις δεδομένων αφορά στη δομή δεδομένων ενός **πίνακα** *(array)*, ο οποίος προκαταλαμβάνει προκαθορισμένο αριθμό θέσεων στην κύρια μνήμη του υπολογιστή. Οι θέσεις αυτές δέχονται δεδομένα του ιδίου τύπου (π.χ. ακέραιοι, πραγματικοί, χαρακτήρες, ακολουθίες χαρακτήρων, εγγραφές) και είναι προσπελάσιμες με τη βοήθεια ενός δείκτη. Στο παράδειγμα που ακολουθεί παρουσιάζεται ένας πίνακας και οι περιπτώσεις ελέγχου οριακών περιπτώσεων τιμών για τον δείκτη του πίνακα **(πίνακας 11.2).**

Ο δείκτης του πίνακα *(array index)* είναι μία μεταβλητή μέσω της τιμής της οποίας μπορούμε να προσπελάσουμε ένα στοιχείο του πίνακα.

Παράδειγμα:

Πίνακας		2	3	4	5	6	7	

Περιπτώσεις ελέγχου οριακών περιπτώσεων δεδομένων			
Περιγραφή Ελέγχου	Τιμή Εισόδου	ΚατάστασηΕξόδου	**Έλεγχος Ορίων του Δείκτη**
Τιμή Δείκτη Ίση με την Αρχική Τιμή Δείκτη	2	Υπαρκτή	Χαμηλός Δείκτης, Εντός ορίων
Τιμή Δείκτη Μικρότερη της Αρχικής Τιμής Δείκτη	1	Σφάλμα	**Χαμηλός Δείκτης, Εκτός ορίων**
Τιμή Δείκτη Ίση με την Τελική Τιμή Δείκτη	7	Υπαρκτή	Υψηλός Δείκτης Εντός ορίων
Τιμή Δείκτη Μεγαλύτερη από την Τελική Τιμή Δείκτη	8	Σφάλμα	**Υψηλός Δείκτης Εκτός ορίων**
Τιμή Δείκτη Εντός των Ορίων του Δείκτη	4	Υπαρκτή	Εντός Ορίων
Τιμή Δείκτη Πολύ Μικρότερη από την Αρχική Τιμή του Δείκτη	-7	Σφάλμα	**Εκτός Ορίων**
Τιμή Δείκτη Πολύ Μεγαλύτερη από την Τελική Τιμή του Δείκτη	20	Σφάλμα	**Εκτός Ορίων**

Πίνακας 11.2. - Περιπτώσεις ελέγχου οριακών περιπτώσεων Δείκτη Πίνακα που λαμβάνει ακέραιες τιμές στο διάστημα 2 έως 8.

Σημείωση: Για πρακτική άσκηση Βλέπε Τετράδιο Εργασίας Κεφάλαιο 11, Μάθημα 1ο, Δραστηριότητα 4.

Ανακεφαλαίωση / περίληψη

Η ύπαρξη σφαλμάτων είναι αναμενόμενη στο οποιοδήποτε έργο παραγωγής λογισμικού. Ο έλεγχος είναι η διαδικασία εκτέλεσης ενός προγράμματος με στόχο την αποκάλυψη των σφαλμάτων του λογισμικού. Μία περίπτωση δεδομένων ελέγχου θεωρείται κατάλληλη για χρήση εάν έχει υψηλή πιθανότητα ανάδειξης σφαλμάτων. Επιτυχής θεωρείται ένας έλεγχος εάν επιτύχει να αναδείξει ένα σφάλμα που δεν ήταν γνωστό. Οι δραστηριότητες ελέγχου του λογισμικού με δοκιμαστικά δεδομένα ολοκληρώνονται όταν η διαδικασία ελέγχων δεν αναδεικνύει σφάλματα ή όταν παρέλθει το χρονικό διάστημα του ελέγχου.

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Να δημιουργήσετε δοκιμαστικά δεδομένα για τον έλεγχο της μονάδας των πράξεων Διαχείρισης Βιβλίων της Εφαρμογής Βιβλιοθήκης.
2. Εφαρμόστε έλεγχο White Box και Black Box για το λογισμικό που αναπτύξατε για την Εφαρμογή Βιβλιοθήκης.
3. Δημιουργείστε πίνακα των σφαλμάτων του λογισμικού που αναπτύξατε για την εφαρμογή Βιβλιοθήκης.
4. Δημιουργείστε πίνακα ελέγχου οριακών περιπτώσεων για το σύστημα μενού της Εφαρμογής Βιβλιοθήκης.

Γλωσσάριο - Ευρετήριο όρων

δοκιμαστικά δεδομένα *(test data)* - Δεδομένα που επιλέγονται με σκοπό τη δοκιμή ενός προγράμματος [217].
αναμενόμενα αποτελέσματα *(expected results)* - Αποτελέσματα με βάση τα οποία είναι δυνατό να ελεγχθεί, εάν η υλοποίηση των λειτουργιών έχει πραγματοποιηθεί σωστά [218].
τύποι σφαλμάτων *(error types)* - Κατάταξη των σφαλμάτων του λογισμικού, ανάλογα με την αιτία που τα προκάλεσε [217].
εκσφαλμάτωση *(debugging)* – Διαδικασία που αποσκοπεί στην ανίχνευση, τον εντοπισμό και τη διόρθωση των σφαλμάτων του κώδικα του λογισμικού [217].
σφάλματα προγραμματισμού *(program errors)* - Σφάλματα που γίνονται από τον προγραμματιστή κατά τη διάρκεια σύνταξης του προγράμματος [218].
έλεγχος οριακών περιπτώσεων δεδομένων *(boundary conditions testing)* - Τύπος ελέγχου που αποκαλύπτει τη συμπεριφορά του προγράμματος όταν διοχετευτούν σε αυτό δεδομένα εισόδου τα οποία βρίσκονται στα όρια του διαστήματος του πεδίου ορισμού των δεδομένων εισόδου [221].

Βιβλιογραφία

Sommerville I., "Software Engineering", Addison-Wesley, 1985
Myers G., "The art of Software Testing", John Wiley & Sons Inc, 1979
Γαρίδης Π., Δεληγιαννάκης Μ., "Σύγχρονο λεξικό Πληροφορικής", Εκδόσεις Δίαυλος, 1989

ΚΕΦΑΛΑΙΟ 12

ΕΝΤΟΠΙΣΜΟΣ ΣΦΑΛΜΑΤΩΝ ΚΑΙ ΠΑΡΑΛΕΙΨΕΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

12.1. Τρόποι και εργαλεία εκσφαλμάτωσης

12.2. Διόρθωση των επιμέρους εργασιών

ΕΙΣΑΓΩΓΗ

Ο David Gelender, καθηγητής Πληροφορικής στο Πανεπιστήμιο Yale, αναφέρει στο καινούργιο του βιβλίο "The Aesthetics of Computing" (εκδόσεις Weidenfeld & Nicolson): "Το κάποτε καινούργιο αεροδρόμιο του Ντένβερ, πέντε χρόνια τώρα περιμένει μάταια το τελικό του λογισμικό για να λειτουργήσει. Ένα αεροδρόμιο, δέκα φορές μεγαλύτερο από το Ελληνικό, δε μπορούσε να αρχίσει να λειτουργεί γιατί δεν ήταν δυνατό στους προγραμματιστές να βρουν που ήταν όλα αυτά τα λάθη στο προγραμματιζόμενο σύστημα της αυτόματης συλλογής και διανομής των αποσκευών, σε ένα δίκτυο με 100 υπολογιστές και 5000 ηλεκτρονικά μάτια επίβλεψης".

- Είναι πράγματι τόσο δύσκολος ο εντοπισμός και η διόρθωση των σφαλμάτων;
- Υπάρχουν διαδικασίες, μέθοδοι και εργαλεία που μπορούν να βοηθήσουν σ' αυτό;
- Ποιος θα εξασφαλίσει ότι η διόρθωση ενός σφάλματος δε θα δημιουργήσει νέα σφάλματα;
- Ποιος θα συντονίσει, θα ελέγξει και θα εποπτεύσει τις τροποποιήσεις στο λογισμικό;

Μερικές απαντήσεις σ' αυτά τα ερωτήματα δίνονται στις ενότητες που ακολουθούν. Άλλωστε, ας μη ξεχνάμε τον ισχυρισμό του DeMacro ότι "το μέσο λογισμικό προϊόν δεν είναι ποτέ απαλλαγμένο σφαλμάτων".

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να διακρίνει τη διαφορά ελέγχου και εκσφαλμάτωσης.
2. Να εκτιμά τις δυσκολίες εντοπισμού σφαλμάτων και να χρησιμοποιεί την πιο κατάλληλη μέθοδο εκσφαλμάτωσης.
3. Να μπορεί να διαχειρίζεται αποτελεσματικά την κάθε τροποποίηση στο λογισμικό.

12.1. ΤΡΟΠΟΙ ΚΑΙ ΕΡΓΑΛΕΙΑ ΕΚΣΦΑΛΜΑΤΩΣΗΣ

Από τη διαδικασία των ελέγχων *(testing)* σ' ένα σύστημα λογισμικού (βλ. κεφάλαιο 10) είναι φυσικό να προκύψουν διάφοροι τύπων σφαλμάτων (βλ. κεφάλαιο 11), δηλαδή αποκλίσεις μεταξύ των αναμενόμενων αποτελεσμάτων και αυτών που προκύπτουν κατά την εκτέλεση των προγραμμάτων. Πως όμως θα εντοπιστεί η αιτία που προκάλεσε το σφάλμα; Ποιές αλλαγές θα πρέπει να γίνουν στο λογισμικό, ώστε να αποκλειστεί η περαιτέρω παρουσία αυτού του σφάλματος; Τη διπλή αυτή διαδικασία, του εντοπισμού δηλαδή της αιτίας του σφάλματος και την εξάλειψή του, αναλαμβάνει η εκσφαλμάτωση *(debugging)*.

Ορισμός

Εκσφαλμάτωση *(debugging)* είναι η διαδικασία που έχει σα στόχο: (α) να εντοπίσει την αιτία κάποιου σφάλματος στο λογισμικό και (β) να επιφέρει τις κατάλληλες αλλαγές στο λογισμικό ώστε να εξαλειφθεί η αιτία του σφάλματος.

Ο αγγλικός όρος για την εκσφαλμάτωση (debugging) σημαίνει κατά λέξη "διώξιμο των κοριών" και προήλθε από ένα πραγματικό γεγονός που συνέβη στα πρώτα χρόνια των υπολογιστών. Στη δεκαετία του '50 οι υπολογιστές ήταν μεγαθήρια, τα οποία αποτελούνταν από χιλιάδες λυχνίες και καταλάμβαναν ολόκληρους ορόφους κτιρίων. Κάποτε έπαθε βλάβη ένας υπολογιστής, στον οποίο εργαζόταν η Grace Murray Hopper, δημιουργός της γλώσσας Cobol. Μετά από έλεγχο πολλών ωρών ανακαλύφθηκε ότι η βλάβη οφειλόταν σε ένα μεγάλο σκώρο (bug), που είχε μπλεχτεί ανάμεσα σε δύο καλώδια και είχε πάθει ηλεκτροπληξία. Μόλις αφαιρέθηκε το έντομο, η μηχανή δούλεψε πάλι κανονικά. Τότε η Hopper επινόησε τον όρο "debugging", ο οποίος επικράτησε μέχρι σήμερα.

Τα λάθη στο λογισμικό διακρίνονται σε **συντακτικά** και **λογικά.** Τα συντακτικά λάθη είναι τα λάθη τα οποία προκαλούνται από την παραβίαση των συντακτικών κανόνων της χρησιμοποιούμενης γλώσσας προγραμματισμού. Έτσι, τα λάθη αυτά μπορεί να οφείλονται σε αναγραμματισμούς των εντολών της γλώσσας (για παράδειγμα, PRNIT αντί PRINT), σε παραλείψεις ή λάθη στις δηλώσεις των μεταβλητών κ.λ.π. Τα συντακτικά λάθη ανιχνεύονται συνήθως από τον μεταγλωττιστή *(compiler)*, ο οποίος και παράγει γι' αυτό κατάλληλα **διαγνωστικά μηνύματα.** Τα λογικά λάθη εμφανίζονται στη διάρκεια εκτέλεσης του προγράμματος και μπορεί να οφείλονται σε σφάλματα κατά την υλοποίηση των αλγορίθμων, σε λανθασμένες αλληλεπιδράσεις ανάμεσα στα διάφορα τμήματα του προγράμματος κ.λ.π.

Για οποιαδήποτε περίπτωση σφάλματος, η εκσφαλμάτωση είναι μία επίπονη διαδικασία, για την η οποία δεν υπάρχει σαφής μεθοδολογία. Βασίζεται σε μεγάλο βαθμό στις αναλυτικές ικανότητες του προγραμματιστή και στην πείρα. Παρόλα αυτά υπάρχουν κάποιες προσεγγίσεις (μέθοδοι) εκσφαλμάτωσης, οι οποίες παρουσιάζονται στην ενότητα 12.1.3. Στο σημείο αυτό πρέπει να τονιστεί ότι ο καλός σχεδιασμός του λογισμικού επιβάλλει την πρόβλεψη των συνθηκών σφάλματος και τον κατάλληλο χειρισμό σε περίπτωση σφαλμάτων, έτσι ώστε να αναδρομολογείται ή να τερματίζεται μία διαδικασία *(process)* όταν συμβαίνει κάποιο σφάλμα. Ο Yourdon ονομάζει αυτή την προσέγγιση **"antibugging".** Η προσέγγιση αυτή προβλέπει τον εμπλουτισμό του προγράμματος με κατάλληλες εντολές χειρισμού σφαλμάτων *(error-handling)*, ώστε αυτό να μη διακόπτεται σε περίπτωση οποιουδήποτε σφάλματος.Δυστυχώς, πολλές φορές ενσωμα-

τώνουμε τον χειρισμό σφαλμάτων στο λογισμικό και στη συνέχεια ποτέ δεν τον ελέγχουμε (σχετικά, κυκλοφορεί το πραγματικό ανέκδοτο που περιγράφεται στο πλαίσιο).

Ένα διαλογικό σύστημα λογισμικού είχε αναπτυχθεί κατόπιν συμβολαίου. Σε μία μονάδα του λογισμικού που πραγματοποιούσε συναλλαγές *(transactions)*, ένας προγραμματιστής αστειευόμενος, και βέβαια σίγουρος για το αλάνθαστο πρόγραμμά του, έβαλε το ακόλουθο μήνυμα χειρισμού λάθους μετά από μια σειρά συνθηκών ελέγχου: "ΣΦΑΛΜΑ! ΔΕΝ ΥΠΑΡΧΕΙ ΚΑΝΕΝΑΣ ΤΡΟΠΟΣ ΝΑ ΦΤΑΣΕΙΣ ΩΣ ΕΔΩ." Τελικά αυτό το μήνυμα σφάλματος ανακαλύφτηκε από τον πελάτη κατά τη διάρκεια εκπαίδευσης των χρηστών.

Σημείωση: Για πρακτική εφαρμογή της παραπάνω προσέγγισης βλέπε δραστηριότητα 1 στο Τετράδιο Εργασίας.

Γιατί, όμως, η εκσφαλμάτωση θεωρείται επίπονη και δύσκολη διαδικασία; Ορισμένα χαρακτηριστικά των σφαλμάτων δίνουν μερικές απαντήσεις στο ερώτημα:

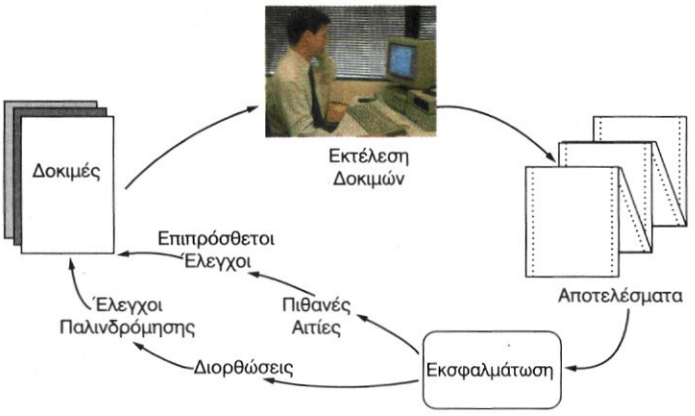
- Το σύμπτωμα και το αίτιο που προκαλεί το σφάλμα είναι πιθανό να βρίσκονται σε διαφορετικά σημεία του προγράμματος. Το σύμπτωμα ίσως εμφανίζεται σ' ένα μέρος του προγράμματος, ενώ η αιτία ίσως να βρίσκεται σ' ένα άλλο απομακρυσμένο σημείο (για παράδειγμα σε εφαρμογές αρχιτεκτονικής εξυπηρετητή - πελάτη *(client - server)*).
- Το σύμπτωμα είναι πιθανό να μην εμφανίζεται προσωρινά, όταν κάποιο άλλο σφάλμα διορθώνεται.
- Το σύμπτωμα είναι πιθανό να μην οφείλεται σε σφάλμα του λογισμικού, αλλά σε άλλους εξωγενείς παράγοντες (όπως π.χ., τα λάθη σε αριθμητικούς υπολογισμούς, που οφείλονται σε στρογγυλοποίηση των δεκαδικών ψηφίων).
- Το σύμπτωμα είναι πιθανό να οφείλεται σε ανθρώπινο λάθος, το οποίο δεν ανιχνεύεται εύκολα.
- Το σύμπτωμα είναι πιθανό να οφείλεται σε πρόβλημα χρονισμού και όχι σε πρόβλημα επεξεργασίας.
- Ίσως είναι δύσκολο να αναπαραχθούν ακριβώς οι συνθήκες εισόδου (για παράδειγμα σε μια εφαρμογή πραγματικού χρόνου *(real time)*, στην οποία η σειρά των δεδομένων εισόδου είναι απροσδιόριστη).
- Το σύμπτωμα είναι πιθανό να οφείλεται σε αιτίες που διαμοιράζονται σε διεργασίες, οι οποίες τρέχουν σε διαφορετικούς επεξεργαστές.
- Οι συνθήκες κάτω από τις οποίες εργάζεται ο προγραμματιστής για να εντοπίσει και να εξαλείψει κάποιο σφάλμα στο λογισμικό, είναι συχνά ιδιαίτερα πιεστικές. Αυτό έχει ως αποτέλεσμα η διόρθωση ενός σφάλματος να προκαλέσει άλλα σφάλματα.

12.1.1. Η διαδικασία της εκσφαλμάτωσης

Όπως φαίνεται στο **σχήμα 12.1,** η διαδικασία της εκσφαλμάτωσης αρχίζει με την εκτέλεση μιας δοκιμής ελέγχου. Τα αποτελέσματα του ελέγχου αξιολογούνται και συγκρίνονται με τα αναμενόμενα αποτελέσματα. Αν διαπιστώσουμε απόκλιση μεταξύ των αναμενόμενων και των πραγματικών αποτελεσμάτων, τότε υπάρχει σφάλμα στο λογισμικό. Συχνά, η μη αντιστοιχία που παρατηρείται μεταξύ αναμενόμενων και πραγματικών αποτελεσμάτων είναι σύμπτωμα μιας βαθύτερης αιτίας, που μέχρι τώρα παραμένει άγνωστη. Με τη διαδικασία της εκσφαλμάτωσης προσπαθούμε να συνδυάσουμε το σύμπτωμα με την αιτία που το προκαλεί και να οδηγηθούμε έτσι στη διόρθωση του σφάλματος.

Η διαδικασία της εκσφαλμάτωσης έχει πάντα μία από τις ακόλουθες εκβάσεις:

- Το αίτιο θα βρεθεί, θα διορθωθεί και θα εξαλειφθεί.



Σχήμα 12.1. - Η διαδικασία της εκσφαλμάτωσης.

- Το αίτιο δεν θα βρεθεί. Σ' αυτή την περίπτωση είναι πιθανό, τα αποτελέσματα του ελέγχου να μας δώσουν κάποιες ενδείξεις για το ποιο θα μπορούσε να είναι το αίτιο που προκαλεί το παρατηρούμενο σφάλμα. Προκειμένου να επιβεβαιώσουμε ή να απορρίψουμε την υποψία μας, σχεδιάζουμε ένα νέο έλεγχο. Τα αποτελέσματα του νέου ελέγχου μπορεί να εντοπίσουν το αίτιο, ή να δώσουν νέες ενδείξεις για τον εντοπισμό του. Τη διαδικασία αυτή επαναλαμβάνουμε όσες φορές χρειαστεί, έως ότου κατορθώσουμε να εντοπίσουμε το αίτιο του σφάλματος.

Ο Brandley, για να μας δείξει τον τρόπο με τον οποίο πρέπει να αναζητούμε την αιτία κάποιου σφάλματος, περιγράφει την εκσφαλμάτωση ως εξής:

"Η εκσφαλμάτωση είναι απλή εφαρμογή μιας επιστημονικής μεθόδου που έχει αναπτυχθεί πριν από 2.500 χρόνια. Η βάση της εκσφαλμάτωσης είναι να εντοπίσει την πηγή του προβλήματος (την αιτία) μέσω υποθέσεων εργασίας, που προβλέπουν νέες τιμές προς εξέταση".
Για να υποστηρίξει την άποψή του, παραθέτει το εξής μη-λογισμικό παράδειγμα από την καθημερινή ζωή:

"Μια λάμπα στο σπίτι δε δουλεύει. Αν τίποτα στο σπίτι δε δουλεύει, η αιτία μπορεί να οφείλεται στο ηλεκτρικό κύκλωμα του σπιτιού ή έξω από αυτό. Κοιτάζουμε έξω να δούμε αν υπάρχει γενικευμένη βλάβη (black out) σε όλη τη γειτονιά. Αν δεν υπάρχει γενικευμένη βλάβη, τοποθετούμε τη λάμπα σ' ένα άλλο φωτιστικό σώμα (για να ελέγξουμε τη λάμπα) και μια καλή ηλεκτρική συσκευή σε μια πρίζα. Με αυτή τη λογική συνεχίζουμε τη διαδοχή των υποθέσεων και κάνουμε τους ελέγχους, που πιστεύουμε ότι θα αποκαλύψουν την αιτία της βλάβης. "

Σχολιάζοντας τις ανθρώπινες πλευρές της εκσφαλμάτωσης, ο Shneiderman (στο βιβλίο του "Software Psychology", Winthrop Publishers, 1980,) αναφέρει: "Η εκσφαλμάτωση είναι ένα από τα πιο εξοργιστικά πράγματα του προγραμματισμού. Έχει στοιχεία επίλυσης προβλήματος ή σπαζοκεφαλιάς που συνδέονται με την ενοχλητική αναγνώριση ότι έχεις κάνει ένα σφάλμα. Η αυξανόμενη ανησυχία και το ότι δε θέλουμε να δεχτούμε την πιθανότητα σφάλματος, αυξάνει τη δυσκολία. Ευτυχώς, υπάρχει η μεγάλη αίσθηση της ανακούφισης και η μείωση της έντασης όταν το σφάλμα τελικά ... διορθώνεται".]

12.1.2. Οι αρχές της εκσφαλμάτωσης

Ο Myers έχει προτείνει μερικές χρήσιμες συμβουλές για την εκσφαλμάτωση, τις οποίες αποκαλεί **"αρχές της εκσφαλμάτωσης"**. Επειδή η εκσφαλμάτωση είναι μια διπλή διαδικασία (εντοπισμός και διόρθωση του σφάλματος), ο Myers προτείνει δύο σύνολα αρχών εκσφαλμάτωσης:

Ο Van Vleck ("Three Questions About Each Bug You Find", ACM Software Engineering Notes, vol. 14, no. 5, July 1989) προτείνει τρεις απλές ερωτήσεις που κάθε μηχανικός λογισμικού θα πρέπει να ρωτάει πριν προχωρήσει στη διόρθωση που θα διώξει την αιτία του σφάλματος:
1. Αναπαράγεται η αιτία του σφάλματος σε κάποιο άλλο τμήμα του προγράμματος;
2. Ποιο είναι το "επόμενο σφάλμα" που μπορεί να δημιουργηθεί από τη διόρθωση που πρόκειται να κάνω;
3. Τι θα μπορούσα να έχω κάνει για να αποτρέψω αυτό το σφάλμα στην πρώτη φάση;

Οι αρχές εντοπισμού του σφάλματος

1. Αν ένα σφάλμα δε μπορεί να αντιμετωπιστεί σε κάποιο εύλογο χρονικό διάστημα (ίσως 30 λεπτά για ένα μικρό πρόγραμμα, ή κάποιες ώρες για ένα μεγάλο), καλό είναι να το "ξεχάσουμε" για λίγο και να ασχοληθούμε με κάτι άλλο. Αργότερα, με καθαρότερη και ξεκούραστη σκέψη, έχομε μεγαλύτερες πιθανότητες να μπορέσουμε να βρούμε τη λύση του.
2. Πολλές φορές μια απλή περιγραφή του προβλήματος σε κάποιον άλλο, οδηγεί στη λύση, χωρίς μάλιστα τη βοήθεια του συνομιλητή.
3. Καλό είναι τα εργαλεία εκσφαλμάτωσης να χρησιμοποιούνται σαν βοηθήματα και όχι σαν υποκατάστατα της σκέψης. Οι έρευνες έχουν δείξει ότι άνθρωποι που αποφεύγουν τέτοια εργαλεία, ακόμα κι αν προσπαθούν να διορθώσουν προγράμματα που δεν οικεία σ' αυτούς, τείνουν να έχουν μεγαλύτερη επιτυχία από τους ανθρώπους που χρησιμοποιούν τέτοια εργαλεία.
4. Το πιο κοινό λάθος που γίνεται, ειδικά από τους άπειρους στην εκσφαλμάτωση, είναι η προσπάθεια να επιλυθεί ένα πρόβλημα κάνοντας πειραματικές αλλαγές στο πρόγραμμα (π.χ., "Δεν ξέρω τι φταίει, θα αλλάξω αυτή την εντολή να δω τι θα συμβεί"). Αυτή η προσέγγιση δεν μπορεί να θεωρηθεί εκσφαλμάτωση. Όχι επειδή έχει μηδαμινή πιθανότητα επιτυχίας, αλλά γιατί συχνά επαυξάνει το πρόβλημα, προσθέτοντας νέα σφάλματα στο πρόγραμμα.

Οι αρχές διόρθωσης του σφάλματος

1. Όπου υπάρχει ένα σφάλμα, είναι πιθανό να υπάρχει κι άλλο. Όταν κάποιος βρίσκει ένα σφάλμα σ' ένα τμήμα ενός προγράμματος, η πιθανότητα ύπαρξης άλλου σφάλματος σ' αυτό το τμήμα είναι μεγαλύτερη. Όταν λοιπόν διορθώνεται κάποιο σφάλμα, καλό είναι να εξετάζεται η γειτονική περιοχή για οτιδήποτε άλλο φαίνεται ύποπτο.
2. Ένα συχνό ελάττωμα είναι η διόρθωση του συμπτώματος του σφάλματος και όχι η διόρθωση του ίδιου του σφάλματος. Αν η προτεινόμενη διόρθωση δεν ταιριάζει με όλες τις ενδείξεις που αφορούν στο σφάλμα, τότε ίσως διορθωθεί μόνο ένα τμήμα του σφάλματος.
3. Η πιθανότητα μια διόρθωση να είναι σωστή δεν είναι 100%. Ο κώδικας που προστίθεται σ' ένα πρόγραμμα για να διορθώσει κάποιο σφάλμα, ποτέ δεν πρέπει να θεωρείται σαν σωστός. Οι επιδιορθώσεις είναι πιο επιρρεπείς σε σφάλματα, από ότι ο αρχικός κώδικας του προγράμματος. Αυτό σημαίνει ότι οι επιδιορθώσεις των σφαλμάτων πρέπει να ελέγχονται πιο αυστηρά απ' ότι το αρχικό πρόγραμμα.
4. Η πιθανότητα μια διόρθωση να είναι σωστή μειώνεται, όσο το μέγεθος του προγράμματος αυξάνει. Η εμπειρία έχει δείξει ότι ο λόγος των σφαλμάτων που οφείλονται σε λανθασμένες επιδιορθώσεις προς τα αρχικά σφάλματα αυξάνει στα μεγάλα προγράμματα. Σε ευρέως χρησιμοποιούμενα μεγάλα προγράμματα, ένα σε κάθε έξι νέα σφάλματα που ανακαλύπτονται είναι σφάλμα που οφείλεται σε προηγούμενη επιδιόρθωση του προγράμματος.
5. Η επιδιόρθωση σφαλμάτων είναι μια μορφή σχεδίασης προγράμματος. Με δεδομένη τη φύση των διορθώσεων, οι οποίες είναι επιρρεπείς σε σφάλματα, η κοινή αίσθηση λέει ότι οι διαδικασίες και μεθοδολογίες που έχουν χρησιμοποιηθεί για τη σχεδίαση, θα πρέπει να εφαρμοστούν και στη διαδικασία διόρθωσης σφαλμάτων.

12.1.3. Μέθοδοι και εργαλεία εκσφαλμάτωσης

Σύμφωνα με τον Myers υπάρχουν τρεις μέθοδοι εκσφαλμάτωσης:

- Άμεσος εξαναγκασμός *(brute force)*.
- Ανίχνευση προς τα πίσω *(backtracking)*.
- Απαλοιφή αιτίας *(cause elimination)*.

Ο **άμεσος εξαναγκασμός** *(brute force)* είναι η πιο κοινή και η λιγότερο αποτελεσματική μέθοδος για την απομόνωση της αιτίας, που προκαλεί ένα λάθος στο λογισμικό. Η φιλοσοφία της μεθόδου αυτής είναι: "άσε τον υπολογιστή να βρει το λάθος". Έτσι, με τη χρήση κατάλληλων εργαλείων λογισμικού που χρησιμοποιούνται στη διαδικασία της εκσφαλμάτωσης, που λέγονται διορθωτές λαθών *(debuggers)*, παίρνονται αποτυπώματα μνήμης *(memory dumps)* και εκτελούνται ανιχνεύσεις ροής *(flow traces)* και έτσι, τα προγράμματα γεμίζουν με εντολές εμφάνισης ενδεικτικών ή πληροφοριακών μηνυμάτων στην οθόνη ή στον εκτυπωτή. Με τη μέθοδο του "άμεσου εξαναγκασμού" ελπίζουμε ότι κάπου στον όγκο της πληροφορίας που παράγεται θα βρούμε μια ένδειξη, που θα οδηγήσει στο αίτιο του λάθους. Τις περισσότερες, όμως, φορές η μέθοδος οδηγεί σε σπατάλη κόπου και χρόνου.

Αποτύπωμα μνήμης *(memory dump)* είναι μια τεχνική, που επιτρέπει τη μεταφορά του συνόλου ή τμήματος του περιεχομένου της κύριας μνήμης σε κάποια συσκευή εξόδου (οθόνη ή εκτυπωτή), ή ακόμη στη βοηθητική μνήμη.

Η **ανίχνευση προς τα πίσω** *(backtracking)* είναι μια άλλη μέθοδος εκσφαλμάτωσης, που μπορεί να χρησιμοποιηθεί επιτυχώς σε μικρά προγράμματα. Αρχίζοντας από το σημείο που ένα σύμπτωμα σφάλματος έχει αποκαλυφθεί, ο πηγαίος κώδικας ανιχνεύεται προς τα πίσω (συνήθως με το χέρι), μέχρι να βρεθεί το σημείο της αιτίας του σφάλματος.

Η τρίτη μέθοδος εκσφαλμάτωσης - **απαλοιφή αιτίας** *(cause elimination)* - εξετάζει τα δεδομένα που σχετίζονται με τα σφάλματα που ανιχνεύτηκαν και τα ταξινομεί σε κατηγορίες. Στη συνέχεια επινοείται μια "υπόθεση αιτίας" και χρησιμοποιούνται οι διάφορες κατηγορίες των δεδομένων για να αποδείξουν, ή να αναθεωρήσουν την υπόθεση. Εναλλακτικά, αναπτύσσεται μια λίστα όλων των πιθανών αιτίων σφαλμάτων και διεξάγονται οι έλεγχοι για να εξαλείψουν κάθε ένα από αυτά. Αν στους αρχικούς ελέγχους υπάρχει ένδειξη ότι η υπόθεση αιτίου είναι αληθινή, τα δεδομένα βελτιώνονται σε μια προσπάθεια να απομονωθεί το σφάλμα.

Κάθε μία από τις παραπάνω μεθόδους εκσφαλμάτωσης μπορεί να συμπληρωθεί (υποβοηθηθεί) από τα εργαλεία εκσφαλμάτωσης *(debugging tools)*. Εργαλεία εκσφαλμάτωσης υπάρχουν πάρα πολλά, όπως είναι οι αναλυτές προγράμματος *(program analysers)*, το διαλογικό περιβάλλον εκσφαλμάτωσης *(interactive dubugging environment)* κ.λ.π. Η χρησιμότητά τους, πάντως, είναι ουσιαστική μόνο μετά από προσεκτική ανάλυση των συνθηκών και "δημιουργική" χρήση τους.

Όσον αφορά στους μεταγλωττιστές *(compilers)*, οι περισσότεροι παρέχουν πολύτιμες πληρο-

φορίες για πολύ συνηθισμένα σφάλματα, όπως:

- Δείκτες πίνακα που υπερβαίνουν τη διάστασή του.
- Αποτελέσματα αριθμητικών υπολογισμών που υπερβαίνουν το μέγεθος των μεταβλητών στις οποίες αποθηκεύονται.
- Μεταβλητές που δεν είναι ορισμένες *(undefined)*, ή που δεν παίρνουν αρχική τιμή από το πρόγραμμα.

Επίσης, πολλοί μεταγλωττιστές συνεργάζονται με πακέτα διόρθωσης (δραστηριότητες 2 και 3 στο τετράδιο του μαθητή), που βέβαια ποικίλουν σε δυνατότητες, αλλά έχουν κάποια κοινά χαρακτηριστικά, όπως είναι:

- Η δυνατότητα λήψης **αποτυπώματος μνήμης** *(memory dump)*.
- Η δυνατότητα λήψης **μεταθανάτιου αποτυπώματος** *(post-mortem dump)*.
- Η δυνατότητα **ιχνηλάτησης ροής** *(flow trace)*.
- Η δυνατότητα εκτέλεσης εντολών του προγράμματος μεταξύ δύο κύριων σημείων, με δυνατότητα επιλογής τιμών από τον προγραμματιστή.

Μεταθανάτιο αποτύπωμα *(post-mortem dump)* είναι μια τεχνική κατά την οποία εκτυπώνονται οι τιμές όλων ή επιλεγμένων μεταβλητών, σε περίπτωση βίαιου (μη φυσιολογικού - *abnormal)* τερματισμού της εκτέλεσης του προγράμματος.

Ιχνηλάτηση ροής *(flow trace)* είναι η τεχνική κατά την οποία καθορίζονται βασικά σημεία *(break points)* του προγράμματος και ζητείται η εκτέλεση του προγράμματος μεταξύ δύο σημείων. Τότε ο διορθωτής *(debugger)* εκτυπώνει πληροφορίες γύρω από τη ροή εκτέλεσης του προγράμματος, σημειώνοντας με ποια σειρά εκτελούνται οι εντολές.

Σύμφωνα με τον Myers, το πρόβλημα των εργαλείων εκσφαλμάτωσης είναι ότι αγνοούν τη διαδικασία του "σκέπτεσθαι". Μελέτες έχουν δείξει ότι τα εργαλεία εκσφαλμάτωσης δε βοηθάνε σε μεγάλο βαθμό τη διαδικασία της εκσφαλμάτωσης και ότι, από άποψη ταχύτητας και ακρίβειας στη εύρεση ενός σφάλματος, οι άνθρωποι που χρησιμοποιούν το μυαλό τους αντί για "βοηθήματα" επιδεικνύουν μεγαλύτερη αποτελεσματικότητα. Τέλος, σύμφωνα με τον Pressman, στην εκσφαλμάτωση ισχύει το εξής ρητό: "Όταν όλα τα άλλα αποτυγχάνουν, ζήτα βοήθεια". Είναι πολύ πιθανό κάποιος άλλος συνάδελφος ή φίλος να εντοπίσει πολύ εύκολα την αιτία κάποιου σφάλματος.

12.2. ΔΙΟΡΘΩΣΗ ΤΩΝ ΕΠΙΜΕΡΟΥΣ ΕΡΓΑΣΙΩΝ

Οι αλλαγές στο λογισμικό είναι αναπόφευκτες κατά τη διάρκεια της ανάπτυξής του, αλλά και μετά την παράδοσή του στον πελάτη, κατά τη διάρκεια της συντήρησης. Κάθε σφάλμα που εντοπίζει ο έλεγχος και που μπορεί να παρουσιαστεί σε κάθε φάση της ανάπτυξης, θα πρέπει να διορθωθεί. Επίσης, πολλές φορές θα πρέπει να γίνουν αλλαγές μετά από απαίτηση του πελάτη, εξ αιτίας νέων αναγκών, ή μετά από αναθεώρηση του προϊόντος κ.λ.π.

Τα συστήματα λογισμικού αποτελούνται συνήθως από μεγάλο αριθμό αλληλοσχετιζόμενων μο-

νάδων *(modules)*. Συχνά δε, στα μεγάλα συστήματα υπάρχουν ανεξάρτητες ομάδες, που εργάζονται συγχρόνως σε διαφορετικές πλευρές του συστήματος. Μερικές φορές οι ομάδες αυτές δε στεγάζονται στον ίδιο χώρο, μπορεί μάλιστα, να εργάζονται ακόμα και σε διαφορετικές πόλεις. Έτσι, μια τροποποίηση μιας μονάδας του προγράμματος που επηρεάζει τη βάση δεδομένων, μπορεί να δημιουργήσει πρόβλημα στην άλλη ομάδα. Ποιος, λοιπόν, θα εξασφαλίσει ότι μια διόρθωση ή μια τροποποίηση σε μια μονάδα δε θα δημιουργήσει νέα σφάλματα; Συνήθως αυτό εξασφαλίζεται με τον **έλεγχο παλινδρόμησης** (βλ. ενότητα 12.2.1). Η διόρθωση, όμως, ενός σφάλματος μπορεί να σημαίνει τροποποίηση πολλών γραμμών του κώδικα, ενημέρωση της τεκμηρίωσης του προγράμματος (βλ. κεφάλαιο 14), αναθεώρηση του σχεδίου ή μεταβολή των απαιτήσεων. Ποιος θα πραγματοποιήσει όλες αυτές τις ενημερώσεις; Ποιος θα ενημερώσει τις άλλες ομάδες ανάπτυξης; Ποιος θα αναλάβει το συντονισμό και τον έλεγχο των αλλαγών; Είναι φανερό ότι οι τροποποιήσεις αυξάνουν το επίπεδο της σύγχυσης μεταξύ των ανθρώπων που εργάζονται σ' ένα έργο. Και η σύγχυση αυξάνεται όταν οι αλλαγές δεν αναλύονται πριν πραγματοποιηθούν, δεν καταγράφονται πριν υλοποιηθούν, δεν αναφέρονται σ' αυτούς που χρειάζεται να γνωρίζουν, ή δεν ελέγχονται μ' ένα τρόπο που θα βελτιώσει την ποιότητα και θα μειώσει τα σφάλματα. Για να μη δημιουργείται, λοιπόν, αυτή η σύγχυση, όλα τα παραπάνω τα αναλαμβάνει η **διαχείριση διάταξης λογισμικού** (βλ. ενότητα 12.2.2).

Μία μεγάλη ανθολογία συζητήσεων στις οποίες αναλύονται οι επιδράσεις των τροποποιήσεων σ' ένα σύνθετο σύστημα λογισμικού, υπάρχει στο βιβλίο των Arnold και Bohner με τίτλο "Software Change Impact Analysis" (IEEE Computer Society Press, 1996).

12.2.1. Έλεγχος παλινδρόμησης

Όπως έχει ήδη αναφερθεί, οι επιτυχημένοι έλεγχοι (κάθε είδους) έχουν σαν αποτέλεσμα την ανακάλυψη σφαλμάτων τα οποία και πρέπει να διορθωθούν. Όποτε, όμως, διορθώνεται το λογισμικό, κάποιες πλευρές των συστατικών του λογισμικού (το πρόγραμμα, η τεκμηρίωση ή τα δεδομένα που το υποστηρίζουν) αλλάζουν. Ο **έλεγχος παλινδρόμησης** *(regression testing)* είναι εκείνη η δραστηριότητα που βοηθάει στο να εξασφαλίσουμε ότι, οποιεσδήποτε αλλαγές δε θα δημιουργήσουν απρόσμενη συμπεριφορά ή επιπρόσθετα σφάλματα. Ο έλεγχος αυτός αναγνωρίζει νέα σφάλματα που εισάγονται, ενώ διορθώνονται τα υπάρχοντα.

Συνήθως ο έλεγχος παλινδρόμησης περιλαμβάνει τρεις διαφορετικές κατηγορίες ελέγχων:
* Ένα αντιπροσωπευτικό δείγμα ελέγχων που θα ελέγξει όλες τις λειτουργίες του λογισμικού.
* Επιπρόσθετους ελέγχους, που εστιάζουν σε λειτουργίες του λογισμικού που είναι πιθανό να έχουν επηρεαστεί από τις αλλαγές.
* Ελέγχους που εστιάζουν στα συστατικά του λογισμικού που έχουν αλλάξει.

Σημείωση: Για πρακτική εφαρμογή των παραπάνω βλέπε δραστηριότητες 4 και 5 στο Τετράδιο Εργασίας.

Ο έλεγχος παλινδρόμησης εκτελείται, για παράδειγμα, σε μια νέα έκδοση, με σκοπό να επαληθεύσει ότι αυτή εκτελεί τις ίδιες λειτουργίες, με τον ίδιο τρόπο που τις εκτελούσε η παλιότερη έκδοση, η οποία είχε ελεγχθεί.

Παράδειγμα:

Έστω ότι σε μια εφαρμογή λογισμικού υπάρχουν οι μονάδες *(modules)* M1 και M2. Η μονάδα M2 περιέχει όλες τις λειτουργίες της μονάδας M1 και κάποιες επιπλέον. Ας υποθέσουμε ότι ο έλεγχος της μονάδας M1 είναι επιτυχής, ενώ στον έλεγχο της μονάδας M2 εντοπίζεται ένα σφάλμα και είναι ανάγκη να τροποποιηθούν μερικές γραμμές του κώδικα. Αν

ακολουθείται η πολιτική του ελέγχου παλινδρόμησης, ο έλεγχος θα πρέπει να περιλαμβάνει τα εξής:

- Τροποποίηση του κώδικα της μονάδας Μ2.
- Έλεγχος των λειτουργιών που επηρεάζονται από την τροποποίηση του κώδικα.
- Έλεγχος των λειτουργιών της μονάδας Μ1, για να επιβεβαιωθεί ότι συνεχίζει να λειτουργεί σωστά.
- Συνέχιση των ελέγχων των λειτουργιών της μονάδας Μ2.

Είναι φανερό ότι όσο προχωράει ο έλεγχος της συνένωσης (σύνθεσης) των τμημάτων του λογισμικού *(integration testing)*, ο αριθμός των ελέγχων παλινδρόμησης μπορεί να μεγαλώσει πάρα πολύ. Γι' αυτό, το σύνολο των ελέγχων παλινδρόμησης πρέπει να σχεδιαστεί έτσι, ώστε να περιλαμβάνει μόνον εκείνους τους ελέγχους, που αφορούν σε μία ή περισσότερες κατηγορίες σφαλμάτων, για κάθε μία από τις βασικές λειτουργίες του προγράμματος. Είναι μη πρακτικό και μάλλον αδύνατο να επανεκτελούνται όλοι οι έλεγχοι για κάθε λειτουργία, κάθε φορά που πραγματοποιείται μια τροποποίηση.

12.2.2. Διαχείριση διάταξης λογισμικού

Διαχείριση διάταξης λογισμικού *(software configuration management)* είναι η διαχείριση και ο έλεγχος όλων των τύπων τροποποιήσεων που μπορεί να υποστεί ένα σύστημα, ώστε να είναι πάντα γνωστή η κατάσταση κάθε συστατικού μέρους του συστήματος. Θα πρέπει κάθε τροποποίηση σε μια μονάδα ή ένα έγγραφο που αφορά τις απαιτήσεις, τον κώδικα, τον έλεγχο ή την τεκμηρίωση να περνά και στα άλλα σημεία που επηρεάζονται. Η διαχείριση διάταξης αρχίζει με το ξεκίνημα του έργου και συνεχίζεται σ' όλη τη διάρκεια της ανάπτυξης, παίζει δε σημαντικό ρόλο, ακόμα και στη συντήρηση. Επειδή τροποποιήσεις μπορούν να συμβούν οποιαδήποτε στιγμή, οι δραστηριότητες της διαχείρισης διάταξης αναπτύσσονται για:

- να αναγνωρίσουν την ανάγκη τροποποίησης,
- να ελέγξουν την τροποποίηση,
- να εξασφαλίσουν ότι η τροποποίηση υλοποιείται κατάλληλα και
- να αναφέρουν τις αλλαγές σε εκείνους που έχουν ενδιαφέρον γι' αυτές.

Ο Babich (στο βιβλίο του "Software Configuration Management", Addison-Wesley, 1986) σχολιάζει: "Η τέχνη του συντονισμού της ανάπτυξης λογισμικού ώστε να μειωθεί η σύγχιση ονομάζεται διαχείριση διάταξης. Διαχείριση διάταξης είναι η τέχνη της αναγνώρισης, της οργάνωσης και του ελέγχου των τροποποιήσεων στο λογισμικό που αναπτύσσεται από την ομάδα ανάπτυξης. Στόχο έχει να μεγιστοποιήσει την παραγωγικότητα και να ελαχιστοποιήσει τα σφάλματα".

Είναι προφανές ότι η επίβλεψη της διαχείρισης διάταξης είναι αρκετά δύσκολη υπόθεση. Ένα σύστημα λογισμικού αποτελείται συνήθως από μεγάλο αριθμό συσχετιζόμενων μονάδων. Όσο πιο πολύπλοκο είναι ένα σύστημα, τόσο περισσότερο επηρεάζεται από μια αλλαγή που γίνεται σε μια απ' αυτές τις μονάδες. Είναι πολύ δύσκολο να παρακολουθούνται όλες οι αλλαγές και οι επιδράσεις τους στα άλλα συστατικά μέρη του συστήματος. Το δύσκολο αυτό έργο της επίβλεψης της διαχείρισης διάταξης αναλαμβάνει η ομάδα διαχείρισης διάταξης. Η ομάδα διαχείρισης διάταξης μπορεί να αποτελείται από ένα πρόσωπο - τον υπεύθυνο έργου - ή από ένα αριθμό προσώπων (αναλυτές, προγραμματιστές, υπεύθυνο βάσεων δεδομένων κ.λ.π.). Η ομάδα αυτή είναι υπεύθυνη να εξασφαλίζει ότι οι αλλαγές θα γίνουν με ακρίβεια και ταχύτητα. Κάθε αλλαγή θα πρέπει πρώτα να εγκριθεί από την ομάδα διαχείρισης, ενώ η τροποποίηση θα πρέπει να περάσει

σε όλη τη σχετική τεκμηρίωση. Επίσης, η ομάδα διαχείρισης θα πρέπει να ειδοποιήσει όλους όσους μπορούν να επηρεαστούν από την αλλαγή.

Από τα παραπάνω γίνεται φανερό πως η ομάδα διαχείρισης διάταξης θα πρέπει να θέτει σε ενέργεια τέτοιες διαδικασίες, οι οποίες θα διαχειρίζονται αποτελεσματικά την κάθε αλλαγή. Το κρίσιμο, συνεπώς, σημείο είναι η εποπτεία των αλλαγών από την ομάδα διαχείρισης διάταξης.

Σημείωση: Για πρακτική εφαρμογή των παραπάνω βλέπε δραστηριότητες 4, 5 και 6 στο Τετράδιο Εργασίας.

Μετά από κάθε τροποποίηση, η ομάδα διαχείρισης διάταξης θα πρέπει να δημιουργεί μια αναφορά της κατάστασης διαχείρισης, στην οποία θα πρέπει να υπάρχουν απαντήσεις στα ερωτήματα:

- Τι συνέβη;
- Ποιος το έκανε;
- Πως συνέβη;
- Τι άλλο θα επηρεάσει;

Η αναφορά αυτή παίζει ζωτικό ρόλο στην επιτυχία της ανάπτυξης ενός μεγάλου έργου λογισμικού. Όταν εμπλέκονται πολλοί άνθρωποι, είναι πολύ πιθανό να συμβεί το σύνδρομο του "δεν γνωρίζει η δεξιά τι ποιεί η αριστερά". Η αναφορά λοιπόν της κατάστασης της διαχείρισης θα βοηθήσει να εξαλειφθούν τα οποιαδήποτε προβλήματα επικοινωνίας μεταξύ των ανθρώπων που εμπλέκονται στο έργο.

Επειδή η διαχείριση διάταξης είναι μεγάλης κρισιμότητας για την επιτυχία ενός μεγάλου συστήματος λογισμικού, πολλά εργαλεία διαχείρισης διάταξης έχουν αναπτυχθεί και χρησιμοποιηθεί τόσο από εκπαιδευτικά ιδρύματα όσο και από μεγάλες επιχειρήσεις. Η χρήση βιβλιοθηκών με πληροφορίες, που επιτρέπουν τον έλεγχο της διαδικασίας αλλαγών, είναι πολύ συνηθισμένη στη διαχείριση διάταξης.

Μερικά από τα γνωστά εργαλεία διαχείρισης διάταξης είναι τα:

MONSTR *(Monitor for Software Trouble Reporting)*: Αυτό επιτρέπει σ' ένα σύνολο αλληλοσχετιζόμενων χρηστών να παρακολουθούν τα αναφερόμενα προβλήματα, αφού καταγράφει όλες τις αναφορές για οποιοδήποτε πρόβλημα και αλλαγή.

SCCS *(Source Code Control System)* και ο διάδοχός του RCS: Τα εργαλεία αυτά (που αναπτύχθηκαν στο Πανεπιστήμιο του Purdue) ελέγχουν όλες τις αλλαγές στον πηγαίο κώδικα, ενώ κάθε τροποποίηση συνοδεύεται από πληροφορίες για το γιατί, πότε, από ποιον και σε ποιο σημείο του κώδικα έγινε η αλλαγή.

Lifespan: Ένα από τα πιο εξεζητημένα εργαλεία διαχείρισης διάταξης, το οποίο και χρησιμοποιεί δικό του σύστημα διαχείρισης βάσης δεδομένων.

Ο Berlack (στο βιβλίο του "Software Configuration Management", Wiley, 1996) παρουσιάζει μια δειγματοληπτική έρευνα σε θέματα διαχείρισης διάταξης λογισμικού, δίνοντας ιδιαίτερη έμφαση στα θέματα των εργαλείων.

12.2.3. Διαχείριση των εκδόσεων λογισμικού

Η διαχείριση των αλλαγών στο λογισμικό υποβοηθείται σε μεγάλο βαθμό από την καλή διαχείριση των εκδόσεων του λογισμικού. Σε κάθε έκδοση δίνεται ένας αριθμός ή ένας κωδικός αναγνώρισης. Όταν η έκδοση τροποποιείται, τότε σε κάθε τροποποιημένη μονάδα δίνεται ένας αριθμός ή κωδικός αναθεώρησης και γίνεται καταγραφή της κάθε μονάδας και έκδοσης. Με αυτό τον τρόπο η ομάδα διαχείρισης της διάταξης μπορεί σε κάθε φάση της ανάπτυξης του λογισμικού να αναγνωρίσει την τρέχουσα έκδοση και τον αριθμό ή τον κωδικό αναθεώρησης της κάθε μονάδας, γνωρίζοντας επίσης τη διαφορά των διαφόρων αναθεωρήσεων. Αυτό είναι πολύ σημαντικό γιατί πολλές φορές η διόρθωση ενός σφάλματος αποτυγχάνει και απαιτείται επαναφορά στην προηγούμενη έκδοση, ώστε να γίνει νέα απόπειρα με νέα προσέγγιση επίλυσης του σφάλματος.

Ανακεφαλαίωση / περίληψη

Ενώ ο έλεγχος σ' ένα σύστημα λογισμικού έχει σκοπό να ανακαλύψει την ύπαρξη κάποιου σφάλματος, η εκσφαλμάτωση (debugging) έχει διπλό στόχο: α) να εντοπίσει την αιτία που προκάλεσε το σφάλμα και β) να επιφέρει τις κατάλληλες αλλαγές στο σύστημα, ώστε να αποκλειστεί η περαιτέρω παρουσία του σφάλματος αυτού.

Η εκσφαλμάτωση είναι μία επίπονη διαδικασία για την οποία δεν υπάρχει σαφής μεθοδολογία. Βασίζεται σε μεγάλο βαθμό στις αναλυτικές ικανότητες του προγραμματιστή και στην πείρα. Παρόλα αυτά ο Myers, μέσω των "αρχών εκσφαλμάτωσης" που έχει διατυπώσει, παρέχει αρκετές συμβουλές σ' αυτούς που πρόκειται να εντοπίσουν ή να διορθώσουν σφάλματα, προτείνοντας τρεις μεθόδους για την εκσφαλμάτωση:

- *Άμεσος εξαναγκασμός (brute force).*
- *Ανίχνευση προς τα πίσω (backtracking).*
- *Απαλοιφή αιτίας (cause elimination).*

*Κάθε μία από αυτές τις μεθόδους εκσφαλμάτωσης μπορεί να συμπληρωθεί (υποβοηθηθεί) από τα **εργαλεία εκσφαλμάτωσης** (debugging tools). Η διόρθωση, όμως, σφαλμάτων και γενικότερα κάθε τροποποίηση στο λογισμικό μπορεί να δημιουργήσει απρόσμενα αποτελέσματα ή επιπρόσθετα σφάλματα. Για να εξασφαλίσουμε ότι δε θα συμβεί κάτι τέτοιο, μετά από κάθε τροποποίηση πραγματοποιείται ο **έλεγχος παλινδρόμησης.** Τέλος, τη διαχείριση και τον έλεγχο όλων των τύπων τροποποιήσεων που μπορεί να υποστεί ένα σύστημα, ώστε να είναι πάντα γνωστή η κατάσταση κάθε συστατικού μέρους, αναλαμβάνει η **διαχείριση διάταξης λογισμικού** (software configuration management).*

Ερωτήσεις – Δραστηριότητες – Θέματα προς συζήτηση

1. Αντιστοιχίστε σωστά τις δύο λίστες. Σε κάθε όρο της αριστερής λίστας αντιστοιχεί μόνον ένας όρος της δεξιάς.

	• διαχείριση εκδόσεων λογισμικού
έλεγχος •	• εντοπισμός και διόρθωση σφαλμάτων
εκσφαλμάτωση •	• ανακάλυψη σφαλμάτων
διαχείριση διάταξης λογισμικού •	• εργαλείο εκσφαλμάτωσης
αναλυτής προγράμματος •	• διαχείριση και έλεγχος τροποποιήσεων
Lifespan •	• πρότυπο για τη διαχείριση διάταξης
	• εργαλείο διαχείρισης διάταξης

2. Απαντήστε με σωστό ή λάθος

α) Προγραμματιστές με ανάλογο επίπεδο εκπαίδευσης και εμπειρίας έχουν την ίδια ικανότητα διόρθωσης σφαλμάτων.

β) Η εκσφαλμάτωση έχει στόχο να βρει και να διορθώσει τα αίτια του σφάλματος στο λογισμικό.

γ) Η ανίχνευση προς τα πίσω βασίζεται στη φιλοσοφία "άσε τον υπολογιστή να βρει το λάθος".

δ) Η προσέγγιση χωρίς σκέψη είναι η πιο αποτελεσματική μέθοδος για την απομόνωση της αιτίας ενός σφάλματος.

ε) Ο έλεγχος παλινδρόμησης αναγνωρίζει τα νέα σφάλματα που δημιουργούνται ενώ διορθώνονται τα υπάρχοντα.

3. Ο κάθε ένας από τους παρακάτω όρους αποτελεί είτε μέθοδο εκσφαλμάτωσης (Μ) είτε τεχνική εκσφαλμάτωσης (Τ); Συμπληρώστε τα κενά με το κατάλληλο γράμμα Μ ή Τ.

α) μεταθανάτιο αποτύπωμα	[....]	β) ανίχνευση ροής	[....]
γ) ανίχνευση προς τα πίσω	[....]	δ) απαλοιφή αιτίας	[....]
ε) αποτύπωμα μνήμης	[....]	στ) προσέγγιση χωρίς σκέψη	[....]

4. Σχολιάστε τις τρεις ερωτήσεις, που ο Van Vleck προτείνει να κάνουμε πριν προχωρήσουμε στη διόρθωση ενός σφάλματος. Τι πλεονέκτημα μας προσφέρει η απάντηση της κάθε ερώτησης; Προσπαθήστε να συσχετίσετε τις τρεις αυτές ερωτήσεις με τις αρχές διόρθωσης του σφάλματος που προτείνει ο Myers.

5. Στην ενότητα 12.1 αναφέρεται ο όρος "antibugging".

α) Προσπαθήστε να αναπτύξετε ένα σύνολο οδηγιών για το antibugging.

β) Συζητήστε τα πλεονεκτήματα και μειονεκτήματα της χρήσης του antibugging.

Γλωσσάριο - Ευρετήριο όρων

απαλοιφή αιτίου *(cause elimination)* - Μέθοδος εκσφαλμάτωσης που εξετάζει τα δεδομένα που σχετίζονται με τα σφάλματα που ανιχνεύτηκαν και τα ταξινομεί σε κατηγορίες [229].

αποτύπωμα μνήμης *(memory dump)* - Τεχνική που επιτρέπει την εκτύπωση των τιμών των μεταβλητών κατά τη διάρκεια εκτέλεσης του προγράμματος [230].

διαχείριση διάταξης λογισμικού *(software configuration management)* - Η διαχείριση και ο έλεγχος όλων των τύπων τροποποιήσεων που μπορεί να υποστεί ένα σύστημα, ώστε να είναι πάντα γνωστή η κατάσταση κάθε συστατικού μέρους του [232].

διορθωτές *(debuggers)* - Εργαλεία λογισμικού που χρησιμοποιούνται στη διαδικασία της εκσφαλμάτωσης [229].

εκσφαλμάτωση *(debugging)* - Διαδικασία που αποσκοπεί στον εντοπισμό και τη διόρθωση των σφαλμάτων του κώδικα ενός προγράμματος [225].

έλεγχος παλινδρόμησης *(regression testing)* - Δραστηριότητα που βοηθάει στο να εξασφαλιστεί ότι οι αλλαγές που γίνονται στον κώδικα ενός προγράμματος δε θα δημιουργήσουν απρόσμενη συμπεριφορά ή επιπρόσθετα σφάλματα [231].
ιχνηλάτηση ροής *(flow trace)* - Τεχνική κατά την οποία καθορίζονται δύο βασικά σημεία στον κώδικα ενός προγράμματος και ζητείται η εκτέλεση του προγράμματος μεταξύ αυτών των δύο σημείων [230].
μεταθανάτιο αποτύπωμα *(post-mortem dump)* - Τεχνική κατά την οποία εκτυπώνονται οι τιμές όλων ή επιλεγμένων μεταβλητών, σε περίπτωση βίαιου τερματισμού της εκτέλεσης του προγράμματος [230].

Βιβλιογραφία

Babich W. A., "Software Configuration Management", Addison-Wesley, 1986.
Bradley J.H., "The science and art of debugging", Computerword August 19, 1985.
De Macro T., "Controlling Software Projects", Yordon Press, 1982.
Myers G., "The art of software testing", Wiley, 1979.
Pressman S. Rogers, "Software Engineering, Fourth Edition", McGraw-Hill Companies Inc., 1997.
Γιακουμάκης Α. Εμμ., "Τεχνολογία Λογισμικού Τόμος Β", Εκδόσεις Α.Σταμούλης, 1994.

ΚΕΦΑΛΑΙΟ 13

ΔΗΜΙΟΥΡΓΙΑ ΤΕΛΙΚΟΥ ΟΛΟΚΛΗΡΩΜΕΝΟΥ ΠΡΟΪΟΝΤΟΣ

ΠΕΡΙΕΧΟΜΕΝΑ

13.1. Οι στόχοι και οι αρχές του τελικού ελέγχου
13.2. Τα στάδια του τελικού ελέγχου
13.3 Εργαλεία ελέγχου
13.4 Τεκμηρίωση ελέγχου

ΕΙΣΑΓΩΓΗ

Στα προηγούμενα κεφάλαια έχουν περιγραφεί όλες οι απαραίτητες ενέργειες για την ανακάλυψη, τον εντοπισμό και τη διόρθωση των σφαλμάτων που υπάρχουν στις μονάδες του προγράμματος *(modules)*, καθώς και στις διεπαφές *(interfaces)* μεταξύ των μονάδων. Έχει ολοκληρωθεί, δηλαδή, ο έλεγχος των μονάδων του προγράμματος *(module testing)*, ο έλεγχος της συνένωσης των μονάδων *(integration testing)* καθώς και η διόρθωση των σφαλμάτων που εντοπίστηκαν κατά τον έλεγχο. Έτσι μέχρι στιγμής, μετά την ολοκλήρωση της σύνθεσης των μονάδων του συστήματος, έχει εξασφαλιστεί ότι ο κώδικας δουλεύει όπως ακριβώς επιδίωκαν οι προγραμματιστές και είναι απαλλαγμένος σφαλμάτων.

- Είναι, όμως, βέβαιο ότι το σύστημα δουλεύει σύμφωνα με τις απαιτήσεις που προσδιορίστηκαν στο στάδιο της ανάλυσης;
- Είναι βέβαιο ότι πληροί τους στόχους του;
- Είναι βέβαιο ότι ικανοποιεί τόσο τους πελάτες όσο και τους κατασκευαστές του; Μπορεί να θεωρηθεί ότι αποτελεί το τελικό προϊόν έτοιμο να παραδοθεί στον πελάτη;

Απαντήσεις στα παραπάνω ερωτήματα μπορούν να δοθούν μόνον αφού ολοκληρωθεί η διαδικασία του τελικού ελέγχου του λογισμικού. Το τι περιλαμβάνει η διαδικασία αυτή αποτελεί το αντικείμενο του παρόντος κεφαλαίου.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ – ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να μπορεί ο μαθητής να εκτιμά την αναγκαιότητα του τελικού ελέγχου στο λογισμικό και να διακρίνει τη διαφορά μεταξύ του τελικού ελέγχου και των ελέγχων που αφορούν στον κώδικα.
2. Να οργανώνει και να εκτελεί αποτελεσματικά τα διάφορα στάδια του τελικού ελέγχου
3. Να επιλέγει τα κατάλληλα εργαλεία ελέγχου και τεκμηριώνει πλήρως τους ελέγχους που έκανε.

13.1. ΟΙ ΣΤΟΧΟΙ ΚΑΙ ΟΙ ΑΡΧΕΣ ΤΟΥ ΤΕΛΙΚΟΥ ΕΛΕΓΧΟΥ

Μόνο όταν δοθεί καταφατική απάντηση στα ερωτήματα που τέθηκαν παραπάνω, στην εισαγωγή του κεφαλαίου, μπορούμε να πούμε ότι το λογισμικό που κατασκευάσαμε αποτελεί και το τελικό προϊόν, έτοιμο να παραδοθεί στον πελάτη ή στον τελικό χρήστη. Συνεπώς, για να θεωρηθεί το λογισμικό που κατασκευάστηκε "τελικό προϊόν" θα πρέπει να επαληθεύσουμε ότι το λογισμικό που αναπτύχθηκε έχει πετύχει τους στόχους του, ικανοποιεί τόσο τους πελάτες όσο και εμάς τους ίδιους και λύνει το πρόβλημα, όπως αυτό προσδιορίστηκε στον καθορισμό των απαιτήσεων.

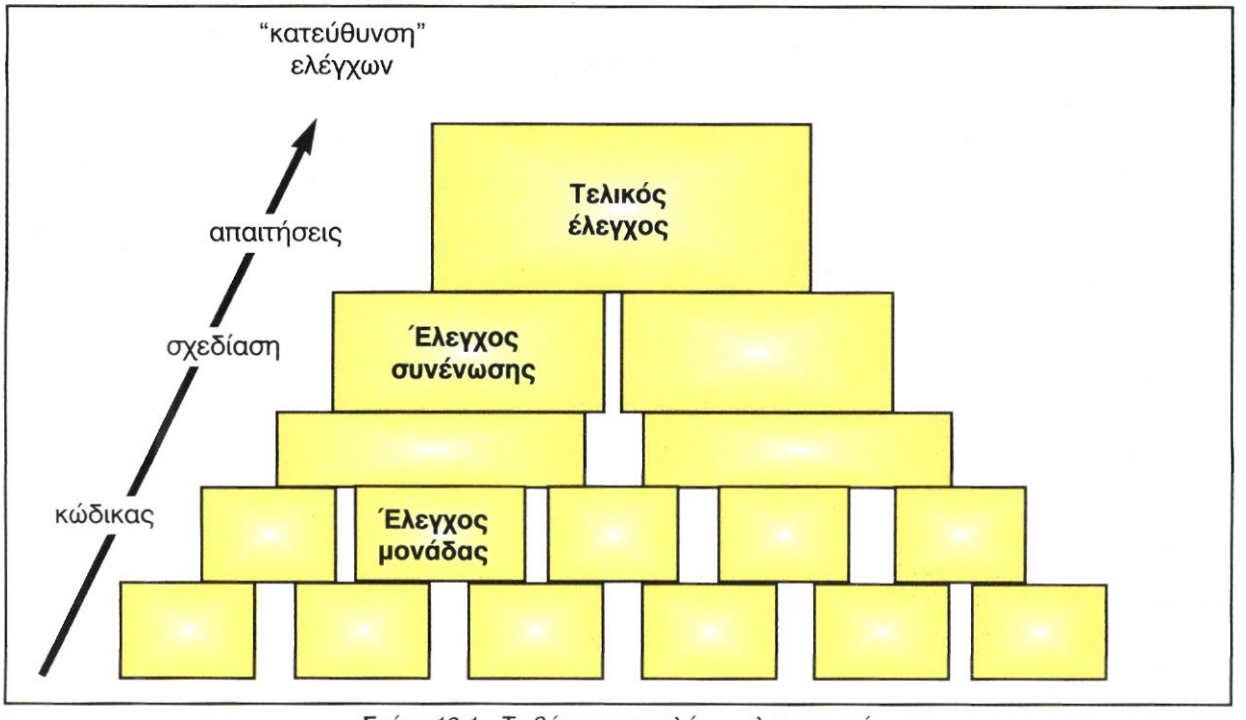
Ο Perry στο βιβλίο του "How to test Software Packages" (Wiley, 1986), παρέχει πρακτικές οδηγίες και χρήσιμους καταλόγους ελέγχου για τον τελικό έλεγχο των πακέτων λογισμικού.

Η διαδικασία των ελέγχων αποτελείται από μια σειρά βημάτων τα οποία υλοποιούνται σειριακά. Τα βήματα αυτά εμφανίζονται στο **σχήμα 13.1.** Αρχικά οι έλεγχοι εστιάζουν στην κάθε μονάδα *(unit)* του λογισμικού (έλεγχος μονάδων), όπως περιγράφεται στο κεφάλαιο 10 (βλ. ενότητα 10.2). Στη συνέχεια οι μονάδες πρέπει να συνενωθούν ώστε να αποτελέσουν το συνολικό λογισμικό πακέτο. Ο έλεγχος συνένωσης *(integration testing)* έχει σκοπό να αντιμετωπίσει τα προβλήματα της επαλήθευσης και της επικύρωσης (βλ. ενότητα 10.2), εντοπίζοντας τις ατέλειες του συστήματος και εξασφαλίζοντας ότι το σύστημα είναι χρήσιμο σε κάθε λειτουργική περίπτωση. Τέλος, μετά τον έλεγχο συνένωσης ακολουθεί ο τελικός έλεγχος. Κατά τον τελικό έλεγχο το λογισμικό πρέπει να συνδυαστεί με τα άλλα στοιχεία του συστήματος (π.χ., υλικό, άνθρωποι, βάσεις δεδομένων), ώστε να επιβεβαιωθεί ότι όλα αυτά τα στοιχεία συνεργάζονται αρμονικά και ότι όλες οι απαιτήσεις (λειτουργικές, απόδοσης κ.λ.π.) του λογισμικού ικανοποιούνται.

Ορισμός

Ο **τελικός έλεγχος** έχει σκοπό να δείξει ότι το όλο σύστημα δουλεύει σύμφωνα με τις λειτουργικές και μη λειτουργικές του απαιτήσεις, πληροί τους στόχους του και ικανοποιεί τον τελικό χρήστη.

Μόνο μετά τον τελικό έλεγχο του προγράμματος μπορούμε να πούμε ότι το λογισμικό που κατασκευάσαμε αποτελεί και το τελικό προϊόν.



Σχήμα 13.1. - Τα βήματα των ελέγχων λογισμικού.

Στο κλασσικό του βιβλίο για τους ελέγχους στο λογισμικό, ο Glen Myers αναφέρει έναν αριθμό κανόνων, που μπορούν να χαρακτηριστούν και σαν στόχοι των ελέγχων:

- Έλεγχος είναι μια διαδικασία εκτέλεσης ενός προγράμματος με σκοπό να βρει κάποιο σφάλμα.
- Μια καλή δοκιμή ελέγχου *(test case)* είναι εκείνη που έχει μεγάλη πιθανότητα να βρει ένα σφάλμα, το οποίο δεν έχει ανακαλυφτεί μέχρι εκείνη τη στιγμή.
- Ένας πετυχημένος έλεγχος είναι εκείνος που ανακαλύπτει ένα σφάλμα, το οποίο δεν έχει εντοπιστεί μέχρι εκείνη τη στιγμή.

Οι στόχοι των ελέγχων, όπως διατυπώνονται παραπάνω, είναι αντίθετοι με την επικρατούσα άποψη ότι, πετυχημένος έλεγχος είναι εκείνος κατά τον οποίο δεν εμφανίζονται σφάλματα. Στόχος μας είναι να σχεδιάζουμε ελέγχους οι οποίοι συστηματικά ανακαλύπτουν διαφορετικές ομάδες λαθών, καταναλώνοντας όσο το δυνατόν μικρότερο χρόνο και προσπάθεια. Αν ο έλεγχος διενεργείται με επιτυχία, θα ανακαλύψει λάθη. Σαν ένα δεύτερο κέρδος, ο έλεγχος αποδεικνύει ότι οι λειτουργίες του λογισμικού εμφανίζονται σύμφωνα με τις προδιαγραφές και ότι οι απαιτήσεις επιδόσεων ικανοποιούνται. Επιπρόσθετα, τα δεδομένα που συλλέγονται όσο διενεργείται ο έλεγχος, παρέχουν μια καλή ένδειξη για την αξιοπιστία του λογισμικού και κάποια ένδειξη για την ποιότητά του (βλ. κεφάλαιο 14).

Πριν προχωρήσουμε στην εξέταση των τελικών ελέγχων, θα πρέπει να κατανοήσουμε τις βασικές αρχές που καθοδηγούν τους ελέγχους λογισμικού, όπως τις προτείνει ο Davis:

- Όλοι οι έλεγχοι θα πρέπει να μπορούν να ανιχνεύουν τις απαιτήσεις του πελάτη. Όπως έχουμε δει, στόχος του ελέγχου είναι να ανακαλύψει σφάλματα. Εξυπακούεται ότι τα πιο σοβαρά

σφάλματα (από την πλευρά του πελάτη) είναι εκείνα που γίνονται η αιτία ώστε να αποτυγχάνει ένα πρόγραμμα να ικανοποιεί τις απαιτήσεις του.

- Οι έλεγχοι θα πρέπει να σχεδιάζονται πολύ πριν από την πραγματοποίησή τους. Η σχεδίαση των ελέγχων μπορεί να αρχίσει αμέσως μόλις οι απαιτήσεις του λογισμικού έχουν καθοριστεί. Σε κάθε περίπτωση, όλοι οι έλεγχοι μπορούν να καταστρωθούν και σχεδιαστούν πριν από την παραγωγή του κώδικα.

Συμπληρωματικά θα μπορούσαμε να αναφέρουμε και τις εξής:

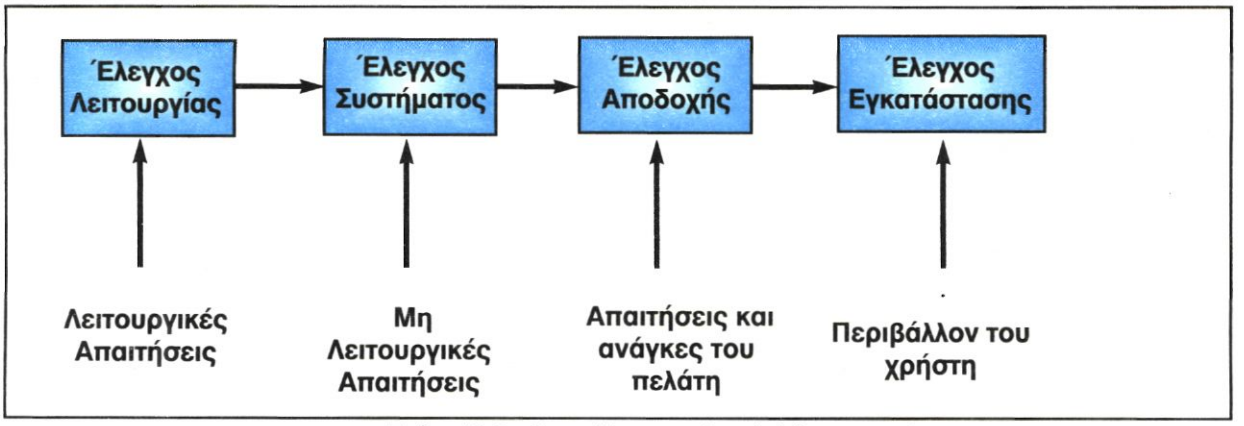
- Η αρχή του Pareto εφαρμόζεται και στους ελέγχους λογισμικού. Το 80% όλων των σφαλμάτων που ανακαλύπτονται κατά τη διάρκεια των ελέγχων είναι πιθανό να είναι ανιχνεύσιμα στο 20% των μονάδων *(modules)* του προγράμματος. Το πρόβλημα, φυσικά, είναι να απομονωθούν αυτές οι ύποπτες μονάδες και να ελεγχθούν εξονυχιστικά.
- Πλήρης έλεγχος δεν είναι δυνατό να γίνει. Ο αριθμός όλων των συνδυασμών για τον έλεγχο ενός μετρίου μεγέθους προγράμματος είναι εξαιρετικά μεγάλος. Γι' αυτό το λόγο είναι αδύνατο να εκτελεστεί κάθε δυνατός συνδυασμός κατά τη διάρκεια των ελέγχων. Είναι, όμως, δυνατό να καλυφτεί ικανοποιητικά η λογική του προγράμματος και να εξασφαλιστεί ότι όλες οι συνθήκες έχουν χρησιμοποιηθεί.
- Για να είναι πιο αποτελεσματικός ο τελικός έλεγχος, θα πρέπει να διενεργείται από μια ανεξάρτητη ομάδα ελέγχου *(independent test group)*. Ενώ η ομάδα ανάπτυξης του λογισμικού είναι υπεύθυνη για τον έλεγχο των ανεξάρτητων μονάδων *(modules)* του προγράμματος (επιβεβαιώνοντας ότι το καθένα εκτελεί τη λειτουργία για την οποία έχει σχεδιαστεί), καθώς και για τον έλεγχο της συνένωσης των μονάδων *(integration testing)*, τους τελικούς ελέγχους τους διενεργεί μια ανεξάρτητη ομάδα ελέγχου. Αυτή η ομάδα ελέγχου αποτελείται από επαγγελματίες ελεγκτές, αναλυτές, σχεδιαστές συστημάτων, ειδικούς σε θέματα διαχείρισης διάταξης και χρήστες. Στην ομάδα ελέγχου δεν εμπλέκονται προγραμματιστές που εργάστηκαν στο έργο, επειδή είναι πολύ εξοικειωμένοι με τη δομή και τους στόχους της υλοποίησης και ίσως έχουν δυσκολία στην αναγνώριση διαφορών μεταξύ της εφαρμογής και της απαιτούμενης λειτουργίας του συστήματος. Κατά τη διάρκεια των ελέγχων από την ομάδα ελέγχου, ο κατασκευαστής του λογισμικού πρέπει να είναι διαθέσιμος, ώστε να διορθώνει τα σφάλματα που η ομάδα ελέγχου ανακαλύπτει.

Σύμφωνα με τον Pressman, υπάρχει ένα πράγμα που ο έλεγχος δε μπορεί να κάνει: "Ο έλεγχος δε μπορεί να δείξει την απουσία ελαττωμάτων, μπορεί μόνο να δείξει ότι υπάρχουν σφάλματα στο λογισμικό".

13.2. ΤΑ ΣΤΑΔΙΑ ΤΟΥ ΤΕΛΙΚΟΥ ΕΛΕΓΧΟΥ

Ο τελικός έλεγχος αποτελείται από τέσσερα στάδια **(σχήμα 13.2)**, τα οποία περιγράφονται αναλυτικά στις ενότητες που ακολουθούν:

- **Έλεγχος λειτουργίας** *(function testing)*: Εξετάζει αν το ολοκληρωμένο σύστημα πληροί τις λειτουργικές απαιτήσεις, όπως αυτές προσδιορίζονται στο έγγραφο απαιτήσεων λογισμικού.
- **Έλεγχος συστήματος** *(system testing)*: Συγκρίνει το σύστημα με τις μη λειτουργικές του απαιτήσεις. Ο έλεγχος αυτός περιλαμβάνει θέματα ασφάλειας, συμβατότητας, όγκου, χρηστικότητας, ακρίβειας και γενικά περιορισμούς κάτω από τους οποίους το σύστημα πρέπει να εκτελεί τις λειτουργίες του.
- **Έλεγχος αποδοχής** *(acceptance testing)*: Συγκρίνει το λογισμικό με τις αρχικές απαιτήσεις και τις τρέχουσες ανάγκες του τελικού χρήστη.
- **Έλεγχος εγκατάστασης** *(installation testing)*: Εντοπίζει τα σφάλματα που γίνονται κατά τη διάρκεια της εγκατάστασης του λογισμικού.



Σχήμα 13.2. - Τα στάδια του τελικού ελέγχου

13.2.1. Έλεγχος λειτουργίας

Το σύνολο των μονάδων του προγράμματος που σχετίζονται με μια λειτουργία ονομάζεται **νήμα (thread).** Γι' αυτό πολλές φορές ο έλεγχος λειτουργίας ονομάζεται και **έλεγχος νήματος.**

Ο **έλεγχος λειτουργίας** *(function testing)* εξετάζει αν το ολοκληρωμένο σύστημα εκτελεί τις λειτουργικές του απαιτήσεις, όπως αυτές προσδιορίζονται στο έγγραφο απαιτήσεων λογισμικού. Για παράδειγμα, ο έλεγχος λειτουργίας ενός λογισμικού διαχείρισης τραπεζικών λογαριασμών επαληθεύει ότι το λογισμικό μπορεί να πιστώσει σωστά μια κατάθεση, να υπολογίσει τους τόκους, να εισάγει μια ανάληψη, να τυπώσει το υπόλοιπο κ.λ.π.

Αυτό το στάδιο ελέγχου αγνοεί τη δομή του συστήματος (εφαρμογής) και εστιάζει στη λειτουργικότητα. Έτσι, για τον έλεγχο λειτουργίας δε χρειάζεται να γνωρίζουμε τι κάνουν τα διάφορα τμήματα του συστήματος, αλλά πρέπει να γνωρίζουμε τις λειτουργικές απαιτήσεις του συστήματος. Πολλές φορές μια λειτουργία υλοποιείται μόνο από κάποια συστατικά μέρη του συστήματος. Σ' αυτή την περίπτωση ο έλεγχος της συγκεκριμένης λειτουργίας μπορεί να αρχίσει, αν υπάρχει λόγος, πριν την ολοκλήρωση ολόκληρου του συστήματος.

13.2.2. Έλεγχος συστήματος

Μετά τον έλεγχο λειτουργίας, η ομάδα ελέγχου είναι πεπεισμένη ότι το λογισμικό εκτελεί όλες τις λειτουργίες του, όπως αυτές έχουν προσδιοριστεί. Στη συνέχεια, ο **έλεγχος συστήματος** *(system testing)* εξετάζει το τρόπο εκτέλεσης αυτών των λειτουργιών, εστιάζοντας στους περιορισμούς κάτω από τους οποίους το σύστημα πρέπει να εκτελεί τις λειτουργίες του. Συγκρίνει δηλαδή το σύστημα με τις μη λειτουργικές του απαιτήσεις. Έτσι, ο έλεγχος συστήματος εξετάζει θέματα σχετικά με τους μηχανισμούς προστασίας της εφαρμογής, τη δυνατότητα επαναφοράς μετά από οποιαδήποτε βλάβη ή σφάλμα, τη δυνατότητα χειρισμού μεγάλου όγκου δεδομένων, τη δυνατότητα επικοινωνίας με άλλα συστήματα, την ευκολία χρήσης κ.λ.π. Για παράδειγμα, σ' ένα σύστημα διαχείρισης τραπεζικών λογαριασμών, ο έλεγχος συστήματος αξιολογεί την ταχύτητα και την ακρίβεια των συναλλαγών, τον χρόνο απόκρισης στα ερωτήματα του χρήστη, την ασφάλεια που παρέχει κ.λ.π.

Ο έλεγχος συστήματος συνήθως περιλαμβάνει τους παρακάτω ελέγχους:

- **Έλεγχος ασφάλειας** *(security testing)*: Έχει σκοπό να επιβεβαιώσει ότι όλοι οι μηχανισμοί προστασίας, που έχουν ενσωματωθεί στο σύστημα, θα το προστατεύσουν αποτελεσματικά. Κατά τη διάρκεια των ελέγχων ασφάλειας, το πρόσωπο που εκτελεί τους ελέγχους παίζει το ρόλο του ανθρώπου που προσπαθεί να διεισδύσει στο σύστημα με οποιονδήποτε τρόπο.
- **Έλεγχος επαναφοράς** *(recovery testing)*: Έχει σκοπό να ελέγξει το πόσο καλά το σύστημα επανέρχεται σε ομαλή λειτουργία, μετά από περιπτώσεις βλάβης διάφορων συσκευών, σφαλμάτων, απώλειας δεδομένων, ή άλλων καταστροφικών καταστάσεων. Ο έλεγχος αυτός αναγκάζει το λογισμικό να αποτύχει με διάφορους τρόπους και στη συνέχεια επιβεβαιώνει ότι το σύστημα μπορεί να επανέλθει σωστά σε κανονική λειτουργία. Αν η επαναφορά είναι αυτόματη (δηλαδή εκτελείται από το ίδιο το σύστημα) τότε ελέγχεται η ορθότητα της επανεκκίνησης, της επαναφοράς δεδομένων, των μηχανισμών σημείων ελέγχου κ.λ.π. Αν η επαναφορά του συστήματος απαιτεί ανθρώπινη παρέμβαση, τότε υπολογίζεται ο μέσος χρόνος της επαναφοράς του συστήματος, για να ελεγχθεί αν βρίσκεται μέσα στα αποδεκτά όρια.
- **Έλεγχος όγκου** *(volume testing)*: Εξετάζει αν το λογισμικό μπορεί να χειριστεί τον όγκο των δεδομένων, όπως προβλέπεται από τις απαιτήσεις. Για παράδειγμα, εξετάζονται πεδία, εγγραφές και αρχεία για να διαπιστωθεί αν τα μεγέθη τους μπορούν να περιλάβουν όλα τα αναμενόμενα δεδομένα. Αν ένα πρόγραμμα χειρίζεται αρχεία, που απλώνονται σε πολλαπλά μέσα (π.χ., δίσκοι), κατά τον έλεγχο δημιουργούνται αρκετά δεδομένα για να αναγκάσουν το πρόγραμμα να μεταπηδήσει από το ένα μέσο στο άλλο.
- **Έλεγχος συμβατότητας** *(compatibility testing)*.Ο έλεγχος συμβατότητας έχει στόχο να διαπιστώσει την συνεργασία του λογισμικού με ένα συγκεκριμένο περιβάλλον υλικού *(hardware)*, λειτουργικού συστήματος, δικτύου κ.λ.π. Συνήθως αυτός ο έλεγχος χρειάζεται όταν το σύστημα επικοινωνεί με άλλα συστήματα μέσω διεπαφών *(interfaces)*, ώστε να εξεταστεί κατά πόσον οι λειτουργίες των διεπαφών εκτελούνται σύμφωνα με τις απαιτήσεις.
- **Έλεγχος διάταξης** *(configuration testing)*. Μερικές κατηγορίες λογισμικού, όπως ένα σύστημα διαχείρισης βάσεων δεδομένων, υποστηρίζουν μια ποικιλία από διατάξεις συσκευών υλικού (για παράδειγμα, τύποι και αριθμός συσκευών εισόδου-εξόδου και γραμμών επικοινωνίας, διαφορετικά μεγέθη μνήμης κ.λ.π.). Συχνά ο αριθμός των πιθανών συνδυασμών είναι πολύ μεγάλος, ώστε να ελεγχθεί το λογισμικό με κάθε μια περίπτωση. Πρέπει, όμως, τουλάχιστον το λογισμικό να ελεγχθεί με κάθε τύπο συσκευής και με την ελάχιστη και μέγιστη διάταξη, όπου προβλέπεται να λειτουργήσει.
- **Έλεγχος πίεσης** *(stress testing)*. Ο έλεγχος αυτός αξιολογεί το σύστημα, όταν αυτό δουλεύει υπό πίεση, στα όριά του, σε μια ορισμένη χρονική περίοδο. Ο έλεγχος πίεσης δεν πρέπει να συγχέεται με τον έλεγχο όγκου. Για να γίνει κατανοητό αυτό, ας αντιστοιχίσουμε τους δύο ελέγχους με την αξιολόγηση μιας δακτυλογράφου. Ο έλεγχος όγκου εξετάζει το κατά πόσον η δακτυλογράφος μπορεί να τα καταφέρει με ένα πολύ μεγάλο κείμενο, ενώ ο έλεγχος πίεσης εξετάζει αν η δακτυλογράφος μπορεί να δακτυλογραφήσει με ένα ρυθμό 50 λέξεων το λεπτό. Οι δοκιμές πίεσης, που πάντα περιλαμβάνουν το στοιχείο του χρόνου, σχεδιάζονται έτσι ώστε να ελεγχθεί η αντίδραση του λογισμικού σε περίπτωση ακραίων καταστάσεων (ταυτόχρονη λειτουργία όλων των συσκευών, ταυτόχρονη προσπέλαση όλων των χρηστών, εκτέλεση μεγάλου αριθμού σύνθετων αναζητήσεων ένα σύστημα διαχείρισης βάσεων δεδομένων κ.λ.π).
- **Έλεγχος χρηστικότητας** *(usability testing)*. Ο έλεγχος αυτός, ο οποίος ονομάζεται και έλεγχος ανθρώπινου παράγοντα *(human-factor)*, εξετάζει τις απαιτήσεις που έχουν σχέση με τη διεπαφή *(interface)* του χρήστη με το σύστημα, όπως οθόνες, εισαγωγή δεδομένων, φόρμες εκτύπωσης, κ.λ.π. Μερικοί από τους παράγοντες που πρέπει να εξεταστούν κατά τον έλεγχο αυτό είναι η ευκολία χρήσης του προγράμματος, η επάρκεια των επιλογών του και το επίπεδο κατανόησης των διαγνωστικών μηνυμάτων, το επίπεδο κατανόησης των εξόδων του προγράμματος.

Ο έλεγχος ασφάλειας πρέπει να εξετάζει την ευπάθεια του συστήματος, τόσο από μετωπικές επιθέσεις, όσο και από πλευρικές ή εκ των όπισθεν επιθέσεις" (Beizer B., "Software System Testing and Quality Assurance", Van Nostrand Reinhold, 1984).

Ο Rubin ("Handbook of Usability testing", Wiley, 1994) έχει γράψει ένα χρήσιμο οδηγό για εκείνους οι οποίοι πρέπει να ασκήσουν τον έλεγχο χρηστικότητας.

ΠΑΡΑΤΗΡΗΣΗ

Τα τελευταία χρόνια παρουσιάζουν μεγάλο ενδιαφέρον οι **γραφικές διεπαφές χρήστη** *(graphical user interfaces ή GUIs)*, των οποίων η πολυπλοκότητα είναι αυξημένη. Αυτό συνεπάγεται δυσκολία στο σχεδιασμό και την εκτέλεση των ελέγχων. Ας δούμε μια σειρά ερωτήσεων που μπορεί να αποτελέσουν οδηγό για μια σειρά ελέγχων που αφορούν τα GUIs, όσον αφορά τα παράθυρα (windows):

➢ Μπορεί το παράθυρο να αλλάζει μέγεθος ή να μετακινηθεί;

➢ Είναι όλες οι λειτουργίες που σχετίζονται με το παράθυρο διαθέσιμες όταν απαιτείται;

➢ Το παράθυρο κλείνει ομαλά;

➢ Μπορούν να επιλεγούν όλα τα περιεχόμενα του παραθύρου από το ποντίκι και τα πλήκτρα λειτουργίας *(function keys)*;

13.2.3. Έλεγχος αποδοχής

Είναι πρακτικά αδύνατο για κάποιον που αναπτύσσει το λογισμικό να προβλέψει πως ακριβώς θα το χρησιμοποιήσει ο τελικός χρήστης. Υπάρχει πιθανότητα να έχουν παρερμηνευτεί οι οδηγίες ή να έχει χρησιμοποιηθεί παράξενος συνδυασμός δεδομένων ή ακόμα η έξοδος (output) των προγραμμάτων να φαίνεται μεν ξεκάθαρη σ? αυτόν που πραγματοποιεί τους ελέγχους, αλλά να μην είναι κατανοητή στον τελικό χρήστη. Είναι λοιπόν απαραίτητο να διενεργηθεί μια σειρά ελέγχων αποδοχής του τελικού προϊόντος από τον πελάτη, κατά τους οποίους εξετάζεται το σύστημα σε σχέση με τις απαιτήσεις του πελάτη.

Ορισμός

Έλεγχος αποδοχής *(acceptance testing)* είναι η διαδικασία σύγκρισης των αρχικών απαιτήσεων του λογισμικού με τις τρέχουσες ανάγκες του τελικού χρήστη. Ο έλεγχος αυτός γίνεται από τον πελάτη ή τον τελικό χρήστη, ο οποίος και καθορίζει ποιες δοκιμασίες ελέγχου θα εξεταστούν.

Υπάρχουν διάφοροι τρόποι με τους οποίους ο πελάτης μπορεί να ελέγξει το σύστημα. Αν το λογισμικό αναπτύσσεται σαν ένα προϊόν με σκοπό να χρησιμοποιηθεί από πολλούς πελάτες, τότε είναι πρακτικά αδύνατο ο έλεγχος αποδοχής να πραγματοποιηθεί από κάθε πελάτη ξεχωριστά. Σ' αυτή την περίπτωση, οι περισσότεροι κατασκευαστές προϊόντων λογισμικού χρησιμοποιούν μια διαδικασία που ονομάζεται **"Άλφα τεστ"** *(alpha test)* και **"Βήτα τεστ"** *(beta test)*, με σκοπό να αποκαλύψουν λάθη που μόνο ο τελικός χρήστης φαίνεται ότι μπορεί να βρει. Η φάση του **"Άλφα τεστ"** γίνεται από ένα πελάτη στους χώρους του κατασκευαστή του λογισμικού. Εκεί ο κατασκευαστής καταγράφει τα λάθη και τα προβλήματα χρήσης που ο πελάτης μπορεί να βρει χρησιμοποιώντας το λογισμικό. Η εκτέλεση του **"Βήτα τεστ"** διενεργείται στους χώρους ενός ή περισσότερων πελατών από τους τελικούς χρήστες. Αντίθετα με τη διαδικασία του "Άλφα τεστ", ο κατασκευαστής συνήθως δεν είναι παρών στην εκτέλεση του "Βήτα τεστ". Γι' αυτό, το "Βήτα τεστ" είναι μια "ζωντανή" εκτέλεση του λογισμικού σ' ένα περιβάλλον που δεν μπορεί να ελεγχθεί από τον κατασκευαστή. Ο πελάτης καταγράφει όλα τα προβλήματα που αντιμετωπίζει κατά τη διάρκεια του "Βήτα τεστ" και τα αναφέρει στον κατασκευαστή σε τακτά χρονικά διαστήματα. Σαν αποτέλεσμα των προβλημάτων που αναφέρονται κατά τη διάρκεια των "Βήτα τεστ", ο κατασκευαστής του λογισμικού κάνει τις απαραίτητες τροποποιήσεις και προετοιμάζει την έκδοση του τελικού προϊόντος, που θα διατεθεί τελικά στους πελάτες.

Μια άλλη κατηγορία ελέγχων αποδοχής είναι οι έλεγχοι απόδοσης *(benchmarking tests)*. Οι έλεγχοι αυτοί χρησιμοποιούνται συνήθως όταν ο πελάτης έχει ειδικές απαιτήσεις. Ο πελάτης ετοιμάζει ένα σύνολο δοκιμών ελέγχου που αντιπροσωπεύουν συνθήκες, κάτω από τις οποίες το σύστημα θα λειτουργήσει όταν εγκατασταθεί. Οι έλεγχοι απόδοσης διενεργούνται από τους πραγματικούς χρήστες ή από μια ειδική ομάδα που γνωρίζει τις λειτουργίες του συστήματος. Σε κάθε περίπτωση, οι ελεγκτές πρέπει να είναι εξοικειωμένοι με τις απαιτήσεις του συστήματος. Εδώ πρέπει να αναφερθεί μια βασική διαφορά μεταξύ ενός ελέγχου απόδοσης και του "Βήτα τεστ". Ενώ ο έλεγχος απόδοσης περιλαμβάνει ένα σύνολο ειδικών δοκιμών ελέγχου που εφαρμόζουν οι χρήστες, το "Βήτα τεστ" βασίζεται στην καθημερινή λειτουργία του συστήματος για να ελέγχονται όλες οι λειτουργίες.

Μια τελευταία μορφή ελέγχου αποδοχής, που όμως ενσωματώνει και τους ελέγχους συμβατότητας και λειτουργίας, είναι και ο παράλληλος έλεγχος *(parallel testing)*. Ο έλεγχος αυτός διενεργείται στην περίπτωση που ένα νέο σύστημα αντικαθιστά ένα υπάρχον. Το νέο σύστημα λειτουργεί παράλληλα με το υπάρχον. Έτσι οι χρήστες έχουν τη δυνατότητα να συγκρίνουν και να αντιπαραβάλλουν τα δύο συστήματα, ενώ ταυτόχρονα εξοικειώνονται και εμπιστεύονται το νέο σύστημα, αφού μπορούν να συγκρίνουν τα αποτελέσματα των δύο συστημάτων.

Για τον έλεγχο αποδοχής μπορεί να χρησιμοποιηθεί ένας συνδυασμός ή και όλες οι παραπάνω προσεγγίσεις. Μετά τον έλεγχο της αποδοχής, ο πελάτης αναφέρει ποιες απαιτήσεις δεν έχουν ικανοποιηθεί και ποιες πρέπει να απαλειφθούν, να τροποποιηθούν ή να προστεθούν εξαιτίας αλλαγής των αναγκών. Αν ο πελάτης μείνει ικανοποιημένος τότε το σύστημα γίνεται αποδεκτό. Καταλήγοντας, πρέπει να πούμε ότι ο έλεγχος αποδοχής αποτελεί την ευκαιρία του πελάτη να επαληθεύσει ότι το λογισμικό που κατασκευάστηκε είναι αυτό που πράγματι επιθυμούσε.

13.2.4. Έλεγχος εγκατάστασης

Ο **έλεγχος εγκατάστασης** *(installation testing)* δεν έχει σκοπό να βρει σφάλματα στο λογισμικό, αλλά να εντοπίσει σφάλματα που γίνονται κατά τη διαδικασία της εγκατάστασης. Αποτελεί την τελευταία μορφή ελέγχου, η οποία βεβαίως δεν είναι πάντα απαραίτητη. Αν ο έλεγχος αποδοχής δεν πραγματοποιήθηκε στους χώρους του χρήστη, ή οι συνθήκες του ελέγχου αποδοχής δεν ήταν ίδιες με τις πραγματικές συνθήκες του χώρου του χρήστη, τότε ο έλεγχος εγκατάστασης είναι απαραίτητος.

Ο έλεγχος εγκατάστασης απαιτεί συνεργασία του κατασκευαστή του λογισμικού με τον πελάτη. Για τον έλεγχο αυτό θα πρέπει να έχει ήδη εγκατασταθεί ο απαιτούμενος εξοπλισμός και να έχουν τεθεί σε λειτουργία οι επικοινωνίες με άλλα συστήματα, αν αυτό είναι απαραίτητο. Θα πρέπει επίσης να προσδιοριστούν τα αρχεία και οι σχέσεις πρόσβασης μεταξύ επικοινωνιών και δεδομένων. Οι δοκιμές ελέγχου εγκατάστασης επιβεβαιώνουν ότι το σύστημα είναι πλήρες και ότι όλα τα απαραίτητα αρχεία και συσκευές υπάρχουν. Επίσης επαληθεύει την ομαλή λειτουργία όλων των στοιχείων του συστήματος που μπορεί να επηρεάζονται από τις συνθήκες του χώρου. Όταν ο έλεγχος είναι πλήρης το σύστημα παραδίδεται επίσημα στον πελάτη.

13.3. ΕΡΓΑΛΕΙΑ ΕΛΕΓΧΟΥ

Υπάρχουν πολλά διαθέσιμα εργαλεία που μπορούν να βοηθήσουν στη μεγάλη ποικιλία δοκιμών που περιλαμβάνει ο τελικός έλεγχος. Τα περισσότερα από αυτά τα εργαλεία είναι αυτοματοποιημένα και συλλέγουν δεδομένα που μπορούν να χρησιμεύσουν στην αξιολόγηση της λειτουργίας και της απόδοσης του συστήματος. Υπάρχουν όμως και εργαλεία, που χρησιμεύουν στην παραγωγή δοκιμασιών ελέγχου και στην καταγραφή των αποτελεσμάτων του ελέγχου.

Μια εξαιρετική πηγή πληροφοριών για αυτόματα εργαλεία ελέγχου λογισμικού είναι το "Testing Tools Reference Guide" (Software Quality Enfineering, Inc., Jacksonville, FL, 1995). Στον κατάλογό του περιέχονται περιγραφές εκατοντάδων εργαλείων ελέγχου, ταξινομημένα ανά είδος ελέγχου, πλατφόρμα υλικού *(hardware)* και είδος υποστήριξης λογισμικού.

Τρία από τα πιο συνηθισμένα εργαλεία ελέγχου είναι ο "παρακολουθητής", ο "αναλυτής" και ο "προσομοιωτής".

Ο **παρακολουθητής** *(monitor)* είναι μια συσκευή, που συλλέγει δεδομένα, τα οποία περνούν από μια συσκευή ή διαδικασία σε μια άλλη. Έτσι, ένας παρακολουθητής εισόδου/εξόδου, αν τοποθετηθεί ανάμεσα σε δύο επεξεργαστές, μπορεί να αποθηκεύει τα δεδομένα που ρέουν μεταξύ των επεξεργαστών. Ο παρακολουθητής μπορεί να καταγράψει ένα στιγμιότυπο των συνθηκών, που επικρατούν πριν και μετά την εμφάνιση ενός σφάλματος. Μ' αυτό τον τρόπο είναι εύκολη η ανίχνευση της πηγής του, καθώς και η επαλήθευση της σωστής εκτέλεσης.
Ένας **αναλυτής** *(analyzer)* όχι μόνο συλλέγει δεδομένα (όπως ο παρακολουθητής), αλλά και τα αξιολογεί, σύμφωνα με προκαθορισμένα κριτήρια. Για παράδειγμα, ένας αναλυτής χρονισμού δουλεύει σε προκαθορισμένες περιοχές μνήμης ή κώδικα και ανιχνεύει πόσος χρόνος δαπανάται σε κάθε περιοχή, καθώς εκτελούνται οι λειτουργίες του συστήματος. Αυτή η ανίχνευση μπορεί να είναι χρήσιμη κατά τον έλεγχο συστήματος, όταν ελέγχεται η συμμόρφωση με τις απαιτήσεις του χρόνου.
Ένας **προσομοιωτής** *(simulator)* παρουσιάζει σ' ένα σύστημα όλα τα χαρακτηριστικά μιας συσκευής ή ενός συστήματος, χωρίς στην πραγματικότητα να έχουμε στη διάθεσή μας τη συσκευή ή το σύστημα. Όπως ένας προσομοιωτής πτήσης μας επιτρέπει να μαθαίνουμε να πετάμε, χωρίς να έχουμε στη διάθεσή μας ένα πραγματικό αεροπλάνο, έτσι κι ένας προσομοιωτής συσκευής μας επιτρέπει να χειριζόμαστε μια συσκευή, ενώ αυτή δεν είναι παρούσα. Ο προσομοιωτής συσκευής είναι ιδιαίτερα χρήσιμος, αν η πραγματική συσκευή βρίσκεται σε χώρο διαφορετικό από αυτόν που γίνεται ο έλεγχος. Μερικές φορές ο προσομοιωτής συσκευής είναι πιο χρήσιμος από την ίδια τη συσκευή, αφού έχει τη δυνατότητα να αποθηκεύει δεδομένα που δείχνουν την κατάσταση της συσκευής κατά τη διάρκεια των διαφόρων σταδίων ελέγχου. Ο προσομοιωτής μπορεί επίσης να παράγει εκθέσεις αναφοράς σχετικές με την κατάστασή του όταν εμφανίζεται σφάλμα, βοηθώντας έτσι στην εξεύρεση της πηγής του σφάλματος. Τέλος, ο προσομοιωτής βοηθάει στον έλεγχο πίεσης και χωρητικότητας, αφού μπορεί να προγραμματιστεί να φορτώσει το σύστημα με σημαντικές ποσότητες δεδομένων και μεγάλο αριθμό χρηστών.

Στον κατάλογο του "Software Quality Engineering" εμφανίζονται οι εξής έξι κατηγορίες εργαλείων ελέγχου:

- Εργαλεία συλλογής δεδομένων: Είναι τα εργαλεία που συλλέγουν δεδομένα που θα χρησιμοποιηθούν κατά τη διάρκεια των ελέγχων.
- Εργαλεία στατικής μέτρησης: Είναι τα εργαλεία που αναλύουν τον πηγαίο κώδικα χωρίς να εκτελείται μια δοκιμασία ελέγχου.

- Εργαλεία δυναμικής μέτρησης: Είναι τα εργαλεία που αναλύουν τον πηγαίο κώδικα κατά τη διάρκεια της εκτέλεσης.
- Εργαλεία προσομοίωσης: Είναι τα εργαλεία που προσομοιώνουν λειτουργίες συστημάτων.
- Εργαλεία διαχείρισης ελέγχου: Είναι τα εργαλεία που βοηθούν στη σχεδίαση, ανάπτυξη και συντονισμό των ελέγχων.
- Διαλειτουργικά εργαλεία: Είναι εργαλεία που ξεπερνάνε τα όρια των παραπάνω κατηγοριών.

13.4. ΤΕΚΜΗΡΙΩΣΗ ΕΛΕΓΧΟΥ

Ο έλεγχος είναι μια πολύ δύσκολη και πολύπλοκη διαδικασία, που απαιτεί μεγάλο συντονισμό, ειδικά στην περίπτωση των μεγάλων συστημάτων. Για το λόγο αυτό υπάρχει ανάγκη μιας πλήρους και καλά σχεδιασμένης τεκμηρίωσης του ελέγχου. Υπάρχουν διάφορα πρότυπα τεκμηρίωσης ελέγχου, που η ομάδα ελέγχου μπορεί να ακολουθεί. Ένα τέτοιο πρότυπο τεκμηρίωσης ελέγχου είναι το ANSI/IEEE Std. 829/1983.

Ανακεφαλαίωση / περίληψη

Για να μπορούμε να πούμε ότι το λογισμικό που κατασκευάστηκε αποτελεί το τελικό προϊόν, θα πρέπει να πραγματοποιηθεί ο τελικός έλεγχος. Ενώ με τους μέχρι τώρα ελέγχους (έλεγχος μονάδας και έλεγχος συνένωσης) και την εκσφαλμάτωση έχει εξασφαλιστεί σε μεγάλο ποσοστό, ότι ο κώδικας δουλεύει όπως ακριβώς επιδίωκαν οι προγραμματιστές (χωρίς σφάλματα), σκοπός του τελικού ελέγχου είναι να δείξει ότι το όλο σύστημα δουλεύει σύμφωνα με τις απαιτήσεις (λειτουργικές και μη), πληροί τους στόχους του και ικανοποιεί τον τελικό χρήστη. Ο τελικός έλεγχος πραγματοποιείται από μια ανεξάρτητη ομάδα, η οποία μπορεί να χρησιμοποιήσει διάφορα αυτοματοποιημένα εργαλεία ελέγχου, τα οποία μπορούν να τη βοηθήσουν στο έργο της. Η ομάδα ελέγχου συνεργάζεται στενά με την ομάδα ανάπτυξης κατά τη διάρκεια των ελέγχων, ενώ είναι απαραίτητη η υιοθέτηση κάποιου προτύπου τεκμηρίωσης ελέγχου.

Ο τελικός έλεγχος αποτελείται από τέσσερα στάδια: από τον έλεγχο των λειτουργικών απαιτήσεων (λειτουργικός έλεγχος), τον έλεγχο των μη λειτουργικών απαιτήσεων (έλεγχος συστήματος), τον έλεγχο αποδοχής και τον έλεγχο εγκατάστασης. Μετά την ολοκλήρωση όλων των παραπάνω ελέγχων το τελικό προϊόν μπορεί να παραδοθεί στον πελάτη.

Ερωτήσεις - Δραστηριότητες - Θέματα προς συζήτηση

1. Βάλτε στη σωστή σειρά τα τέσσερα στάδια του τελικού ελέγχου:
   α. έλεγχος αποδοχής β. έλεγχος συστήματος
   γ. έλεγχος εγκατάστασης δ. έλεγχος λειτουργίας

2. Απαντήστε με σωστό ή λάθος
   α) Πετυχημένος έλεγχος είναι εκείνος κατά τον οποίο δεν εμφανίζονται σφάλματα.
   β) Τους τελικούς ελέγχους τους διενεργεί η ομάδα ανάπτυξης του λογισμικού.
   γ) Οι έλεγχοι πρέπει να σχεδιάζονται πολύ πριν από την πραγματοποίησή τους.
   δ) Ο προσομοιωτής είναι ένα εργαλείο εκσφαλμάτωσης.
   ε) Ο έλεγχος συστήματος συγκρίνει το σύστημα με τις λειτουργικές του απαιτήσεις.

3. Συμπληρώστε τις παρακάτω φράσεις
   α) Ο έλεγχος λειτουργίας ονομάζεται και έλεγχος .....
   β) Ο παράλληλος έλεγχος είναι μια μορφή του ελέγχου .....
   γ) Ο έλεγχος συστήματος συγκρίνει το σύστημα με τις μη ..... του απαιτήσεις.
   δ) Ο έλεγχος πίεσης είναι μια μορφή του ελέγχου .....
   ε) Το benchmarking test είναι μια κατηγορία του ελέγχου .....

4. Επιλέξτε ένα συγκεκριμένο GUI μιας εφαρμογής με την οποία είστε εξοικειωμένοι και σχεδιάστε μια σειρά ελέγχων ώστε να το εξετάσετε.

5. Προσπαθείστε να καταγράψετε μερικά προβλήματα που συνδέονται με τη δημιουργία της ανεξάρτητης ομάδας ελέγχου.

Γλωσσάριο - Ευρετήριο όρων

δοκιμή "άλφα" *(alpha test)* - Μια μορφή ελέγχου αποδοχής, που διενεργείται από τον πελάτη στους χώρους του κατασκευαστή του λογισμικού [246].
δοκιμή "βήτα" *(beta test)* - Μια μορφή ελέγχου αποδοχής που διενεργείται από τους τελικούς χρήστες στο χώρο του πελάτη [246].
έλεγχος απόδοσης *(benchmarking test)* - Μια μορφή ελέγχου αποδοχής που περιλαμβάνει ειδικές δοκιμές και διενεργείται από τους πραγματικούς χρήστες [246].
έλεγχος αποδοχής *(acceptance testing)* - Στάδιο του τελικού ελέγχου, που συγκρίνει το λογισμικό με τις αρχικές απαιτήσεις και τις τρέχουσες ανάγκες του τελικού χρήστη [246].
έλεγχος εγκατάστασης *(installation testing)* - Στάδιο του τελικού ελέγχου, που εντοπίζει τα σφάλματα τα οποία γίνονται κατά τη διάρκεια της εγκατάστασης του λογισμικού [247].
έλεγχος λειτουργίας *(function testing)* - Στάδιο του τελικού ελέγχου, που εξετάζει αν το ολοκληρωμένο σύστημα πληροί τις λειτουργικές του απαιτήσεις [244].
έλεγχος συστήματος *(system testing)* - Στάδιο του τελικού ελέγχου, που συγκρίνει το ολοκληρωμένο σύστημα με τις μη λειτουργικές του απαιτήσεις [244].
παράλληλος έλεγχος *(parallel testing)* - Μια μορφή ελέγχου αποδοχής που διενεργείται στην περίπτωση που ένα νέο σύστημα αντικαθιστά ένα υπάρχον [247].

Βιβλιογραφία

Beizer B., "Software System Testing and Quality Assurance", Van Nostrand Reinhold, 1984.
Davis A., "201 Principles of Software Development", McGrow-Hill, 1995.
Kaner G., Falk J. and Nguyen H. Q., "Testing Computer Software", 2nd edition, Van Nostrabd Reinhold, 1993.
Myers G., "The Art of Software Testing", Wiley, 1979.

ΚΕΦΑΛΑΙΟ 14

ΤΕΚΜΗΡΙΩΣΗ - ΑΞΙΟΛΟΓΗΣΗ

ΠΕΡΙΕΧΟΜΕΝΑ

14.1. Τι είναι τεκμηρίωση
14.2. Αξιολόγηση ενός λογισμικού προϊόντος
14.3. Παράδοση του προϊόντος στους χρήστες

ΕΙΣΑΓΩΓΗ

Σύμφωνα με τον Pressman, δεν είναι ασυνήθιστο σε μια εταιρία ανάπτυξης λογισμικού να δαπανά το 20% ή 30% από όλη την προσπάθεια ανάπτυξης στην κατασκευή τεκμηρίωσης. Ο Dennis, σ' ένα άρθρο του στο "Computer World", αναφέρει ότι "οι περισσότεροι άνθρωποι συμφωνούν ότι η ποιότητα της τεκμηρίωσης για τον τελικό χρήστη συνεπάγεται την επιτυχία ή την αποτυχία ενός νέου προϊόντος λογισμικού. Τα εγχειρίδια είναι αυτά που βλέπουν πρώτα οι πελάτες και από αυτά διαμορφώνουν άποψη για όλο το πακέτο λογισμικού". Εξ άλλου, ο Bailey, στο βιβλίο του για τα ανθρώπινα σφάλματα στα συστήματα υπολογιστών αναφέρει ότι το 30% των ανθρώπινων σφαλμάτων αποδίδονται άμεσα στην κακής ποιότητας τεκμηρίωση.

- Είναι λοιπόν τόσο σημαντικός ο ρόλος της τεκμηρίωσης στο λογισμικό;
- Σε τι συνίσταται η τεκμηρίωση;
- Ποια είναι εκείνα τα εγχειρίδια τεκμηρίωσης που θα πρέπει να συνοδεύουν το κάθε προϊόν λογισμικού κατά την παράδοσή του στον πελάτη;
- Πώς πρέπει να εκπαιδεύονται οι χρήστες;
- Πως αξιολογείται ένα προϊόν λογισμικού;

Αυτά είναι μερικά από τα ερωτήματα στα οποία απαντά το παρόν το κεφάλαιο.

ΔΙΔΑΚΤΙΚΟΙ ΣΤΟΧΟΙ - ΑΝΑΜΕΝΟΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Να περιγράφει ο μαθητής τη σπουδαιότητα και την αναγκαιότητα της τεκμηρίωσης.
2. Να μπορεί να αξιολογεί το λογισμικό και να χρησιμοποιεί κατάλληλα κριτήρια αξιολόγησης.
3. Να αναφέρεται στα εγχειρίδια τεκμηρίωσης, με οποία πρέπει να συνοδεύεται ένα προϊόν λογισμικού κατά την παράδοσή του στον πελάτη.
4. Να αναγνωρίζει τη σημασία της εκπαίδευσης των τελικών χρηστών.

14.1. ΤΙ ΕΙΝΑΙ ΤΕΚΜΗΡΙΩΣΗ

14.1.1. Ορισμός και η σημασία της τεκμηρίωσης

Ορισμός

Με τον όρο τεκμηρίωση εννοούμε το σύνολο του γραπτού υλικού που περιγράφει τα συστατικά μέρη και τις λειτουργίες ενός νέου συστήματος ή/και τις τροποποιήσεις που έγιναν σε ένα υπάρχον σύστημα.

Η τεκμηρίωση λογισμικού *(software documentation)* είναι βασικό συστατικό οποιουδήποτε λογισμικού πακέτου και εξυπηρετεί ένα μεγάλο αριθμό στόχων. Εφοδιάζει τους χρήστες με παραδείγματα και εξηγήσεις για το πως να χρησιμοποιούν το λογισμικό. Περιγράφει τους αλγόριθμους, τις μεθόδους και τις τεχνικές που οι προγραμματιστές έχουν χρησιμοποιήσει για τη δημιουργία του λογισμικού. Οι αναλυτές συστημάτων και οι σχεδιαστές χρησιμοποιούν τεύχη τεκμηρίωσης για να μεταδώσουν τις απαιτήσεις του συστήματος και το σχεδιασμό του στους χρήστες, στη διοίκηση και στις ομάδες υλοποίησης, κατά τη διάρκεια της διαδικασίας ανάπτυξης του λογισμικού.

Από τα παραπάνω γίνεται φανερό ότι η τεκμηρίωση δε συνιστά μια ξεχωριστή φάση στη ζωή ενός συστήματος (όπως εσφαλμένα πολλοί θεωρούν ακόμη και σήμερα, με συνέπεια να την αφήνουν για μετά την ολοκλήρωση του νέου συστήματος). Η τεκμηρίωση δημιουργείται σταδιακά κατά τη διάρκεια των διαφόρων φάσεων ανάπτυξης και μάλιστα σήμερα διατίθενται εργαλεία και τεχνικές που βοηθούν σημαντικά προς αυτή την κατεύθυνση.

Η τεκμηρίωση ενός συστήματος είναι ζωτικής σημασίας για τους παρακάτω λόγους:

- Έργα που αναβάλλονται μπορεί να είναι δύσκολο να ξαναρχίσουν, αν δεν υπάρχει ικανοποιητική τεκμηρίωση, που να θέτει το πρόβλημα, το αντικείμενο, το σκοπό του έργου, καθώς και το βαθμό πληρότητάς του.
- Αν δεν υπάρχει τεκμηρίωση προηγούμενης εργασίας, θα πρέπει να επαναληφθεί μια πλήρης νέα μελέτη ή προσπάθεια πλήρους επαναπρογραμματισμού, προκειμένου να κατανοηθεί η όλη προσπάθεια, πριν αποφασιστεί να γίνει οποιαδήποτε αλλαγή.
- Χωρίς ικανοποιητική τεκμηρίωση, κάθε μετατροπή από ένα σύστημα σε άλλο μπορεί να προκαλέσει σχετική καθυστέρηση ή ένα επιπλέον κόστος.
- Χωρίς αναλυτική και πλήρη τεκμηρίωση είναι δύσκολο να επιτύχει ο έλεγχος του συστήματος.

Ο Brockman δίνει τον εξής ορισμό για την τεκμηρίωση λογισμικού: "Τεκμηρίωση λογισμικού είναι η πληροφορία που έχει σχεδιαστεί με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν."

Ο Brockman αναφέρει ότι τρεις είναι οι βασικοί στόχοι της τεκμηρίωσης:

1. Να αυξήσει την αποτελεσματικότητα. Οι άνθρωποι χρειάζεται να αντιλαμβάνονται πλήρως το σύστημα στο οποίο εργάζονται. Και φυσικά αυτοί δεν μπορούν να εξαρτώνται από πληροφορίες που μεταδίδονται από στόμα σε στόμα, γιατί το προσωπικό αλλάζει συχνά στα περισσότερα περιβάλλοντα ανάπτυξης λογισμικού.
2. Να υπερνικήσει τους φόβους των χρηστών του λογισμικού. Η επιτυχημένη τεκμηρίωση οδηγεί σε επιτυχημένη πρώτη αντιμετώπιση του λογισμικού και επομένως σε μεγαλύτερη αποδοχή και χρήση.
3. Να πουλήσει το προϊόν. Σ' ένα άρθρο του στο "ComputerWorld" (1981) ο Denis αναφέρει: "Οι περισσότεροι άνθρωποι συμφωνούν ότι η ποιότητα της τεκμηρίωσης για τον τελικό χρήστη συνεπάγεται την επιτυχία ή την αποτυχία ενός νέου προϊόντος λογισμικού. Τα εγχειρίδια είναι αυτά που οι πελάτες βλέπουν πρώτα - αυτά σχηματοποιούν τη γνώμη του για όλο το πακέτο λογισμικού".

Για την καλύτερη και πληρέστερη σύνταξη των εγγράφων τεκμηρίωσης έχουν αναπτυχθεί κάποια πρότυπα τεκμηρίωσης *(documentation standards)*. Τα πρότυπα αυτά αποτελούν οδηγούς για τη συγγραφή της τεκμηρίωσης και καθορίζουν το σκοπό, το περίγραμμα και το περιεχόμενο κάθε βασικού εγγράφου τεκμηρίωσης, περιγράφοντας και τα ποιοτικά του χαρακτηριστικά. Επιδίωξη κάθε πρότυπου είναι να παράγει έγγραφα που είναι σαφή, κατανοητά, πλήρη, και εύκολα τροποποιήσιμα.

Πρότυπα τεκμηρίωσης έχει αναπτύξει το IEEE (Institute of Electrical and Electronic Engineers), όπως για παράδειγμα τα:

- ANSI/IEEE std. 830-1984 (έγγραφο περιγραφής απαιτήσεων λογισμικού).
- ANSI/IEEE std.1016-1987 (έγγραφο περιγραφής σχεδίου λογισμικού).
- ANSI/IEEE std. 829-1983 (έγγραφο τεκμηρίωσης ελέγχου λογισμικού)..

Εξέτασε ένα εγχειρίδιο χρήστη *(user manual)* για μία εφαρμογή την οποία χρησιμοποιείς συχνά. Μπορείς να βρεις τουλάχιστον ένα λάθος στην τεκμηρίωση;

Τα λάθη στην τεκμηρίωση μπορεί να είναι τόσο καταστροφικά όσον αφορά στην αποδοχή του προγράμματος, όσο τα λάθη των δεδομένων ή του πηγαίου κώδικα. Δεν είναι τίποτα πιο εξοργιστικό από το να ακολουθεί κάποιος τον οδηγό του χρήστη *(user guide)* και να παίρνει αποτελέσματα που δεν συμπίπτουν με αυτά που προβλέπονται από την τεκμηρίωση. Για το λόγο αυτό ο έλεγχος της τεκμηρίωσης αποτελεί ένα σημαντικό μέρος στον τελικό έλεγχο του λογισμικού.

Ο **έλεγχος τεκμηρίωσης** *(documentation testing)* έχει σκοπό να επιβεβαιώσει κατ' αρχήν ότι έχουν γραφεί όλα τα απαραίτητα εγχειρίδια τεκμηρίωσης (βλ. ενότητα 14.1.2). Στη συνέχεια ελέγχεται το ευανάγνωστο και η ακρίβεια των περιεχομένων. Σε περίπτωση που στις απαιτήσεις προσδιορίζεται η μορφή της τεκμηρίωσης, τότε ο έλεγχος αφορά και την συμμόρφωση μ' αυτές τις απαιτήσεις.

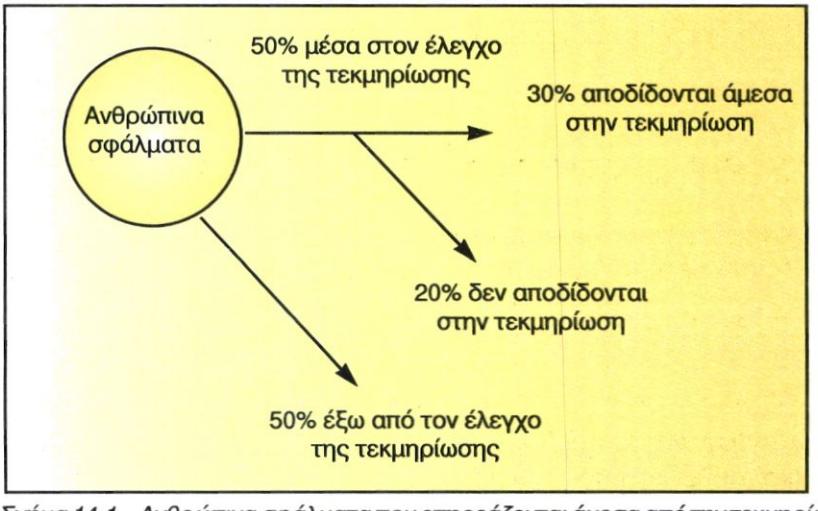
Στο ερώτημα "τι είναι ποιοτικό λογισμικό", ο Philip Crossly (1979), στο βιβλίο του που απετέλεσε ορόσημο σε θέματα ποιότητας, έθεσε τον εξής προβληματισμό: "Το πρόβλημα της ποιότητας δεν είναι τι οι άνθρωποι δεν γνωρίζουν γι? αυτό. Το πρόβλημα είναι τι νομίζουν ότι γνωρίζουν".

Το πιο σημαντικό τμήμα του ελέγχου αφορά την ανίχνευση των προγραμμάτων δια μέσου της τεκμηρίωσης:

- Η τεκμηρίωση περιγράφει με ακρίβεια πως να πραγματοποιηθεί κάθε περίπτωση χρήσης;
- Τα παραδείγματα είναι ακριβή;
- Η ορολογία, οι περιγραφές των menus και οι αποκρίσεις του συστήματος συμφωνούν με το κανονικό πρόγραμμα;
- Είναι σχετικά εύκολο να εντοπιστούν οι οδηγίες στην τεκμηρίωση;
- Μπορεί να γίνει εύκολα η επίλυση των προβλημάτων με τη χρήση της τεκμηρίωσης;
- Εμφανίζονται στην τεκμηρίωση όλα τα μηνύματα λαθών;
- Είναι οι πίνακες περιεχομένων και τα ευρετήρια ακριβή και πλήρη;

Η ανεπαρκής τεκμηρίωση μπορεί να αυξήσει σε μεγάλο βαθμό τα ανθρώπινα λάθη στα συστήματα υπολογιστών. Ο Bailey (1981), στο βιβλίο του για τα ανθρώπινα λάθη στα συστήματα υπολογιστών, αναφέρει ότι το 50% των σφαλμάτων είναι έξω από τον έλεγχο των σχεδιαστών. Το άλλο 50% των ανθρώπινων σφαλμάτων οφείλεται σε αυτούς που σχεδιάζουν και αναπτύσσουν τα συστήματα, και από αυτά το 60% (δηλαδή 30% συνολικά) επηρεάζονται άμεσα από την ποιότητα

της τεκμηρίωσης: εκπαίδευση, γραπτές οδηγίες, διεπαφή *(interface)* ανθρώπου-υπολογιστή επηρεάζονται από την ποιότητα της τεκμηρίωσης (βλ. σχήμα 14.1).



Σχήμα 14.1. - Ανθρώπινα σφάλματα που επηρεάζονται άμεσα από την τεκμηρίωση.

14.1.2. Κατηγορίες τεκμηρίωσης

Μπορούμε να ταξινομήσουμε την τεκμηρίωση του λογισμικού:

- με βάση το περιβάλλον στο οποίο χρησιμοποιείται,
- με βάση το ακροατήριο στο οποίο απευθύνεται και
- με βάση τη φάση ανάπτυξης του λογισμικού.

Όπως οι μελέτες έχουν δείξει, "ένας ικανοποιημένος πελάτης μιλάει σε οκτώ ανθρώπους, ενώ ένας μη ικανοποιημένος μιλάει σε 22 ανθρώπους" (Favin, 1988)

Η ταξινόμηση της τεκμηρίωσης με βάση το περιβάλλον στο οποίο χρησιμοποιείται προτάθηκε από το Αμερικάνικο Ινστιτούτο Επαγγελματικής Εκπαίδευσης *(American Institute of Professional Education, 1981)* και διακρίνει δύο κατηγορίες τεκμηρίωσης: την εξωτερική τεκμηρίωση και την εσωτερική τεκμηρίωση, όπως φαίνεται στον πίνακα που ακολουθεί.

ΤΑΞΙΝΟΜΗΣΗ ΤΗΣ ΤΕΚΜΗΡΙΩΣΗΣ ΜΕ ΒΑΣΗ ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΣΤΟ ΟΠΟΙΟ ΧΡΗΣΙΜΟΠΟΙΕΙΤΑΙ	
Εξωτερική τεκμηρίωση	Είναι η τεκμηρίωση που απευθύνεται έξω από το περιβάλλον του οργανισμού ή της εταιρείας στην οποία αναπτύσσεται το λογισμικό.
Εσωτερική τεκμηρίωση	Είναι η τεκμηρίωση που αναπτύσσεται με σκοπό να χρησιμοποιηθεί μέσα στον ίδιο τον οργανισμό ή την εταιρεία ανάπτυξης του λογισμικού.

Η ταξινόμηση της τεκμηρίωσης με βάση το ακροατήριο στο οποίο απευθύνεται διακρίνει τέσσερις βασικές κατηγορίες, οι οποίες αναλύονται στον παρακάτω πίνακα:

ΤΑΞΙΝΟΜΗΣΗ ΤΗΣ ΤΕΚΜΗΡΙΩΣΗΣ ΜΕ ΒΑΣΗ ΤΟ ΑΚΡΟΑΤΗΡΙΟ ΣΤΟ ΟΠΟΙΟ ΑΠΕΥΘΥΝΕΤΑΙ	
Τεκμηρίωση για τη διοίκηση	Είναι γραπτό υλικό το οποίο πρέπει να τονίζει τα πραγματικά πλεονεκτήματα του νέου συστήματος ως προς την αύξηση της παραγωγικότητας, ή των κερδών του οργανισμού και οφείλει να περιλαμβάνει: ➢ επισκόπηση του συστήματος, ➢ στόχους και προοπτικές του συστήματος, ➢ παραδείγματα από καταστάσεις και οθόνες, που αποτελούν κλειδιά στη διαδικασία λήψης απόφασης, ➢ τελικά κόστη έναντι προϋπολογισμένων, * τελικά χρονοδιαγράμματα έναντι προϋπολογισμένων.
Τεκμηρίωση για το χρήστη	Έχει σκοπό να διδάξει στο προσωπικό πώς να χρησιμοποιεί το νέο σύστημα. Για χάρη των χρηστών συντάσσεται ένα εγχειρίδιο χρήσης *(user's manual)*, το οποίο περιλαμβάνει όλες τις απαραίτητες πληροφορίες που χρειάζονται οι χρήστες για να εκτελέσουν αποτελεσματικά την εργασία τους. Για την τεκμηρίωση για το χρήστη θα αναφερθούμε αναλυτικά στην ενότητα 14.3.2.
Τεκμηρίωση προγράμματος	Προορίζεται για τους προγραμματιστές, οι οποίοι και αναπτύσσουν αυτή την τεκμηρίωση τις περισσότερες φορές. Η καλή τεκμηρίωση του προγράμματος βοηθάει τους άλλους προγραμματιστές στη γρήγορη εκμάθηση του συστήματος, κάτι που είναι απαραίτητο σ' ένα μη σταθερό περιβάλλον ανθρώπων και εξωτερικών συνθηκών, οι μεταβολές των οποίων πρέπει αμέσως να περάσουν στο σύστημα.
Τεκμηρίωση λειτουργίας	Απευθύνεται στο προσωπικό εκμετάλλευσης και χειρισμού και εξηγεί τον τρόπο με τον οποίο πρέπει να εκτελεστεί η ροή των προγραμμάτων, παραθέτοντας τους κανόνες ασφάλειας, εφεδρείας, ελέγχου, προετοιμασίας, εκτέλεσης επανεκκίνησης, διανομής των εξόδων και βοηθητικές λειτουργίες. Έτσι, στα μεγάλα συστήματα υπάρχει η ανάγκη ενός εγχειρίδιου χειρισμού (αναλυτικά βλ. στην ενότητα 14.3.2), παρότι στις ημέρες μας η εξάπλωση των συστημάτων άμεσης επικοινωνίας *(on-line)* και πραγματικού χρόνου *(real-time)* έχουν μειώσει τις καθημερινές χρονοβόρες διαδικασίες, όπως τοποθέτηση μαγνητικών ταινιών στους οδηγούς τους κ.λ.π.

Η ταξινόμηση της τεκμηρίωσης που έχει σαν βάση τη φάση ανάπτυξης του λογισμικού περιλαμβάνει τις κατηγορίες του παρακάτω πίνακα:

ΤΑΞΙΝΟΜΗΣΗ ΤΗΣ ΤΕΚΜΗΡΙΩΣΗΣ ΜΕ ΒΑΣΗ ΤΗ ΦΑΣΗ ΑΝΑΠΤΥΞΗΣ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ	
Τεκμηρίωση απαιτήσεων	Αναπτύσσεται στη φάση του προσδιορισμού των απαιτήσεων του λογισμικού και συνήθως αποτελείται από τρία έγγραφα: • Το έγγραφο ορισμού απαιτήσεων *(requirements definition document)*, που καταγράφει το σκοπό του υπό ανάπτυξη συστήματος, τις λειτουργίες του, το περιβάλλον ανάπτυξης και τους περιορισμούς επιδόσεων. • Το προκαταρκτικό εγχειρίδιο χρήσης *(preliminary user's manual)*, που περιγράφει το πώς θα λειτουργεί το σύστημα μετά την ολοκλήρωση της ανάπτυξής του. Το έγγραφο αυτό απευθύνεται στους χρήστες και σκοπό έχει να τους βοηθήσει να διατυπώσουν τις παρατηρήσεις τους πριν τη σχεδίαση. • Το έγγραφο προσδιορισμού απαιτήσεων *(requirements specification document)*, που προσδιορίζει το τι κάνει το σύστημα, την εξωτερική συμπεριφορά του και τους περιορισμούς υλοποίησης.
Τεκμηρίωση σχεδίου	Είναι το έγγραφο που περιγράφει το λογισμικό που θα αναπτυχθεί. Το έγγραφο αυτό περιέχει την αρχιτεκτονική και τη λεπτομερή σχεδίαση του λογισμικού και είναι απαραίτητο για την κωδικοποίηση, τον έλεγχο και τη συντήρηση του λογισμικού.
Τεκμηρίωση προγράμματος	Είναι το σύνολο των περιγραφών που συνοδεύουν τα προγράμματα, με σκοπό να εξηγήσουν στον αναγνώστη τους τι κάνουν και πως το κάνουν.
Τεκμηρίωση ελέγχου	Έχει σκοπό να ελέγξει την πολυπλοκότητα και τη δυσκολία που υπάρχει κατά τον έλεγχο του λογισμικού από τις ομάδες ελέγχου
Τεκμηρίωση χρήσης και χειρισμού	Αναλυτική αναφορά υπάρχει στην ενότητα 14.3.2
Τεκμηρίωση συστήματος	Αναλυτική αναφορά υπάρχει στην ενότητα 14.3.3

14.1.3. Τεκμηρίωση άμεσης επικοινωνίας

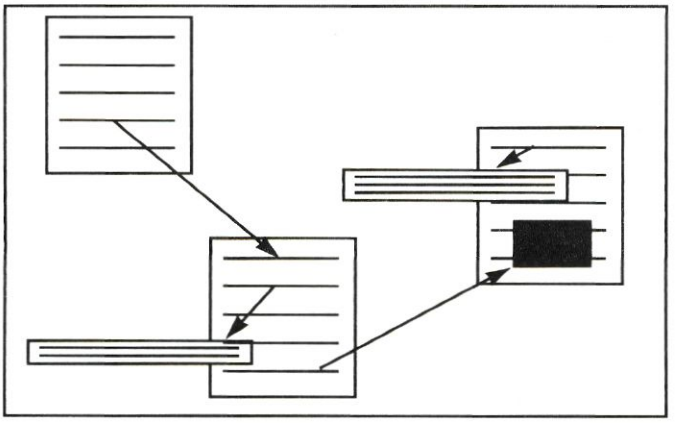
Σύμφωνα με τον Brockman:

"**Τεκμηρίωση άμεσης επικοινωνίας λογισμικού** *(on-line documentation)* είναι η πληροφορία που έχει σχεδιαστεί να παρουσιάζεται στην οθόνη, με σκοπό να διευκολύνει την αλ-

Η τεκμηρίωση άμεσης επικοινωνίας ξεπερνάει τα όρια των παραδοσιακών εγχειριδίων τεκμηρίωσης γιατί ενσωματώνει κείμενο, γραφικά, κίνηση, ηχητικά εφφέ και οργανωτικά σχήματα βάσεων δεδομένων.

ληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν".

Η τεκμηρίωση άμεσης επικοινωνίας είναι ένα βασικό συστατικό στη σύγχρονη ανάπτυξη λογισμικού. Η μεγάλη ανάπτυξη των δικτύων υπολογιστών έχει δώσει τη δυνατότητα της ηλεκτρονικής δημοσίευσης της τεκμηρίωσης σε συστήματα όπως το World Wide Web. Ένα από τα πιο ισχυρά χαρακτηριστικά της τεκμηρίωσης άμεσης επικοινωνίας είναι η δυνατότητα να χρησιμοποιεί **έγγραφα υπερκειμένου** *(hyper-documents)*. Ένα έγγραφο υπερκειμένου επιτρέπει στο χρήστη να έχει πρόσβαση στην πληροφορία με δυναμικό τρόπο. Μια σελίδα της τεκμηρίωσης μπορεί να περιέχει **συνδέσμους υπερκειμένου** *(hypertext links)*, που μπορούν να μεταφέρουν αυτόματα τον χρήστη σε πολλές άλλες σελίδες ή άλλα κείμενα. Έτσι, ο χρήστης μπορεί να ακολουθεί τις συνδέσεις που τον ενδιαφέρουν και να μετακινείται προς τα εμπρός ή προς τα πίσω αναζητώντας μια πληροφορία. Μια τυπική σελίδα υπερκειμένου έχει τη δομή που απεικονίζεται στο **σχήμα 14.2.**



Σχήμα 14.2. - Δομή τυπικής σελίδας υπερκειμένου.

Η τεκμηρίωση άμεσης επικοινωνίας λύνει ορισμένα προβλήματα που αντιμετωπίζουμε με την έντυπη τεκμηρίωση. Τα σημαντικότερα από αυτά είναι τα εξής:

- Στατική οργάνωση: Τα εγχειρίδια από χαρτί έχουν μια συγκεκριμένη στατική οργάνωση: πίνακα περιεχομένων, κεφάλαια, υποκεφάλαια, ευρετήρια κ.λ.π. Έτσι δεν μπορούν να συμπεριλάβουν και να ικανοποιήσουν τις ανάγκες όλων των χρηστών. Αντίθετα η τεκμηρίωση άμεσης επικοινωνίας μπορεί να χρησιμοποιεί βοήθεια από την οθόνη του υπολογιστή, συνδέσμους *(links)* κ.λ.π.
- Όγκος: Τα εγχειρίδια τεκμηρίωσης συνήθως καταλαμβάνουν μεγάλο όγκο. Αυτό δημιουργεί πρόβλημα και στον κατασκευαστή (πρόβλημα αποθήκευσης κυρίως), αλλά και στο χρήστη, αφού ο μεγάλος όγκος τείνει να "τρομοκρατήσει" το χρήστη και να κάνει το εγχειρίδιο "δύσκολο για διάβασμα".
- Ο παθητικός ρόλος των αναγνωστών: Τα έντυπα εγχειρίδια τεκμηρίωσης θεωρείται ότι ενισχύουν τον παθητικό ρόλο του χρήστη. Οι περισσότεροι χρήστες ανατρέχουν στα εγχειρίδια, μό-

νο όταν αντιμετωπίσουν κάποιο πρόβλημα. Αντίθετα, σύμφωνα με τον Jones, "το ενδιαφέρον και η προσοχή του χρήστη αυξάνεται, όταν όλη η πληροφορία της τεκμηρίωσης βρίσκεται σε ένα μέρος και αυτό το μέρος είναι η διεπαφή *(interface)* με το χρήστη".

Η χρησιμοποίηση τεκμηρίωσης άμεσης επικοινωνίας από τους κατασκευαστές λογισμικού έχει αυξηθεί σε μεγάλο βαθμό τα τελευταία χρόνια και συνεχώς αυξάνεται. Στις ημέρες μας, οι μεγαλύτερες εταιρείες παραγωγής λογισμικού προσφέρουν την τεκμηρίωση σε CD-ROMs, ενώ ζητούν επιπρόσθετα χρήματα για την παροχή τεκμηρίωσης σε έντυπα *(paper documentation)*. Δύο είναι οι βασικοί λόγοι γι' αυτό:

- Με τη μεγάλης κλίμακας ψηφιακή αποθήκευση δεδομένων των CD-ROMs, των μαγνητικών και οπτικών δίσκων και άλλων συσκευών μαζικής αποθήκευσης, τα δεδομένα μπορούν να αποθηκευτούν και να διανεμηθούν με χαμηλότερο κόστος, απ' ότι στο χαρτί.
- Η ζήτηση για έγκαιρη, με ακρίβεια και στα μέτρα του πελάτη τεκμηρίωση - κάτι που ξεπερνάει τις δυνατότητες του χαρτιού - συνεχίζει να αυξάνει.

Εκτός των όσων αναφέρθηκαν παραπάνω, υπάρχουν και άλλες βασικές διαφορές μεταξύ της τεκμηρίωσης άμεσης επικοινωνίας και της έντυπης τεκμηρίωσης. Κυριότερες είναι οι εξής :

- **Φυσικές διαφορές:** Ο χώρος στην οθόνη είναι μικρότερος από το χώρο μιας σελίδας Α4 χαρτιού. Σε μια σελίδα χαρτιού χωράνε 200-500 λέξεις, ενώ σε μια οθόνη 50-100 λέξεις. Βέβαια, αυτές οι διαφορές μπορούν να καλυφθούν με ένα διαφορετικό τρόπο γραψίματος της τεκμηρίωσης.
- **Οργανωτικές διαφορές:** Χρησιμοποιώντας τεκμηρίωση άμεσης επικοινωνίας είναι πολύ δύσκολο να βλέπουμε ταυτόχρονα στην οθόνη την τεκμηρίωση και την οθόνη της εφαρμογής, με την οποία εργαζόμαστε. Αντίθετα ένα εγχειρίδιο μπορεί να είναι ανοικτό στο γραφείο μας, ενώ εργαζόμαστε με την εφαρμογή. Επίσης η πλοήγηση στην άμεσης επικοινωνίας τεκμηρίωση είναι μόνο ιεραρχική και παράλληλα δύσκολη, αφού όλες οι οθόνες φαίνεται να ανήκουν στο ίδιο επίπεδο.
- **Άλλες διαφορές:** Ενώ σ' ένα εγχειρίδιο ο χρήστης ξέρει την αρχή και το τέλος, στην τεκμηρίωση άμεσης επικοινωνίας είναι δύσκολο να ξέρει το πόσο έχει διαβάσει και το πόσο θέλει ακόμα να διαβάσει.

14.2. ΑΞΙΟΛΟΓΗΣΗ ΕΝΟΣ ΛΟΓΙΣΜΙΚΟΥ ΠΡΟΪΟΝΤΟΣ

Τα όσα μέχρι τώρα έχουν αναφερθεί σ' αυτό το βιβλίο είχαν σα στόχο την παραγωγή "καλού" λογισμικού. Τι είναι όμως εκείνο που διακρίνει το "καλό" από το "κακό" λογισμικό; "Καλό" λογισμικό είναι εκείνο το λογισμικό που η Τεχνολογία Λογισμικού χαρακτηρίζει ποιοτικό λογισμικό. Τι εννοούμε όμως με τις έννοιες **"ποιοτικό λογισμικό"** και "ποιότητα λογισμικού"; Το Λεξικό Αμερικάνικης Κληρονομιάς (*American Heritage Dictionary)* δίνει τον εξής ορισμό στην ποιότητα *(quality) :* "Ποιότητα είναι ένα χαρακτηριστικό ή μια ιδιότητα κάποιου πράγματος". Κατά συνέπεια, σαν ιδιότητα, η ποιότητα αφορά σε μετρήσιμα (άμεσα ή έμμεσα) χαρακτηριστικά. Τα χαρακτηριστικά του ποιοτικού λογισμικού εξαρτώνται από το ποιος εξετάζει το λογισμικό. Οι χρήστες κρίνουν το λογισμικό σαν υψηλής ποιότητας, αν αυτό κάνει ό,τι αυτοί θέλουν να κάνει με τρόπο εύκολο στη μάθηση και τη χρήση. Το λογισμικό πρέπει επίσης να κρίνεται από αυτούς που το σχεδιάζουν και γράφουν τον κώδικα, καθώς επίσης και από αυτούς που θα το συντηρούν μετά την ολοκλή-

ρωση της ανάπτυξης. Επομένως (σύμφωνα με τον Γιακουμάκη) ένα λογισμικό μπορεί να θεωρηθεί ότι είναι υψηλής ποιότητας αν:

- κάνει ότι οι χρήστες θέλουν να κάνει,
- χρησιμοποιεί τους πόρους του υπολογιστή σωστά και αποδοτικά,
- είναι εύκολο για τον χρήστη να το μάθει και να το χρησιμοποιήσει,
- οι ειδικοί μπορούν να σχεδιάσουν, να κωδικοποιήσουν, να ελέγξουν και να συντηρήσουν το σύστημα σχετικά εύκολα.

Μέχρι σήμερα έχουν προταθεί διάφορες προσεγγίσεις που αφορούν στους παράγοντες εκείνους που καθορίζουν την ποιότητα του λογισμικού. Μεταξύ αυτών γνωστότερη θεωρείται η προσέγγιση του Boehm,

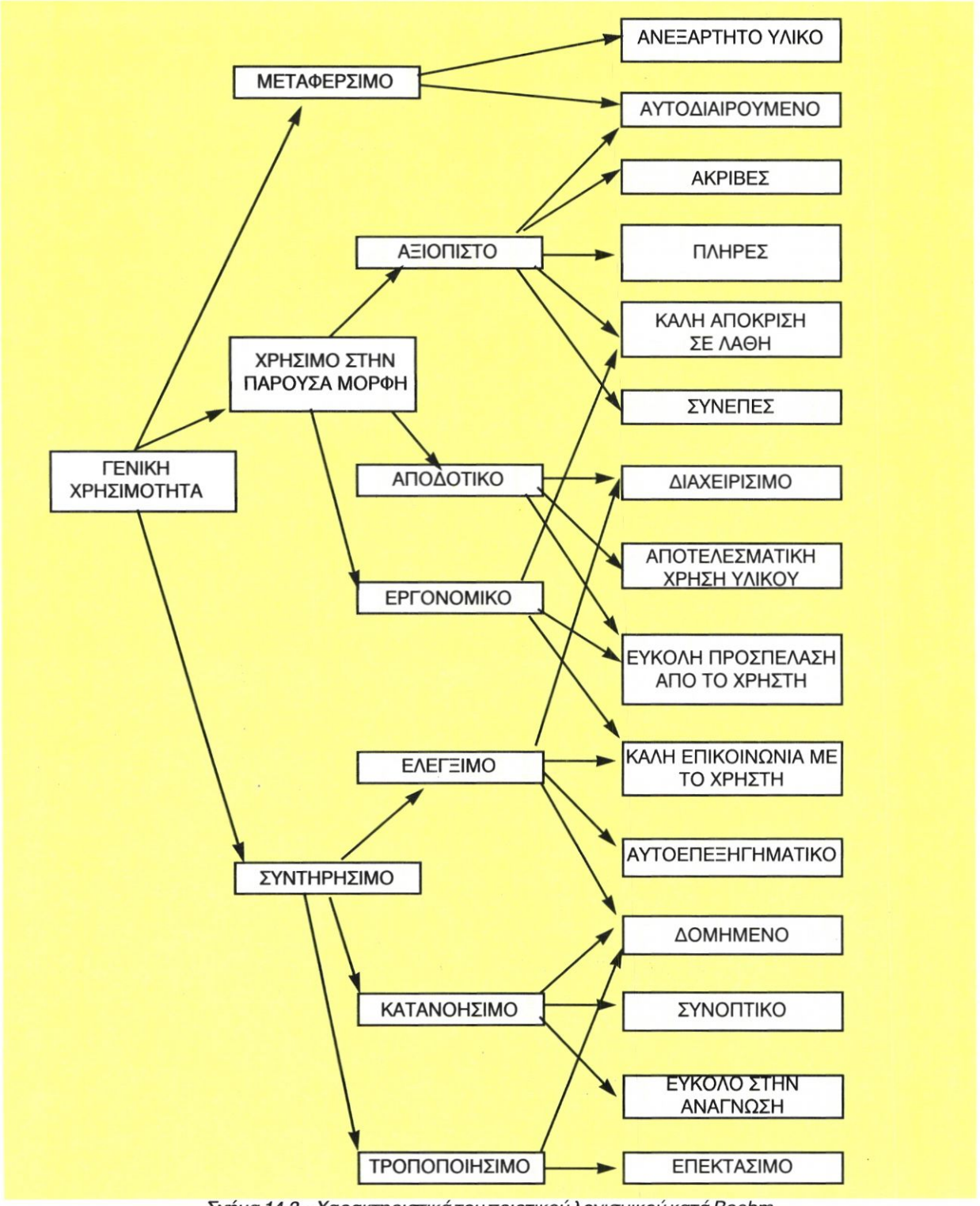
14.2.1. Η προσέγγιση του Boehm

Ο Boehm (1978) αντιμετωπίζει την ποιότητα του λογισμικού σαν μια συγχώνευση συγκεκριμένων χαρακτηριστικών. Τα χαρακτηριστικά αυτά καθώς και η μεταξύ τους σχέση αποτυπώνονται στο **σχήμα 14.3.** Οι έννοιες που δίνει ο Boehm δεν αφορούν μόνο στην επίδοση του λογισμικού αλλά και στις ανάγκες και επιθυμίες των χρηστών.

Επιχειρώντας μια σύντομη πλοήγηση στο **σχήμα 14.3,** φαίνεται ότι, πρώτα απ' όλα, το λογισμικό πρέπει να είναι **χρήσιμο.** Αν δεν είναι, τότε η ανάπτυξή του δεν είναι παρά χαμένος χρόνος, χαμένοι πόροι και άσκοπη προσπάθεια. Μπορούμε να θεωρήσουμε ότι η χρησιμότητα ενός συστήματος διακρίνεται με τρεις τρόπους που αντιστοιχούν σε τρεις κατηγορίες χρηστών (με την ευρεία έννοια) που ενδιαφέρονται για το σύστημα.

Στην πρώτη κατηγορία χρηστών ανήκουν εκείνοι που επιθυμούν να χρησιμοποιήσουν το σύστημα σε άλλους υπολογιστές ή άλλους χώρους. Σε αυτήν την περίπτωση το σύστημα πρέπει να είναι **μεταφέρσιμο** *(portable),* έτσι ώστε να μπορεί να μεταφερθεί από υπολογιστή σε υπολογιστή (ανεξάρτητο από το υλικό). Μερικές φορές η συνολική διάταξη του συστήματος (υλικό και λογισμικό) παραμένει η ίδια, αλλά ένα τμήμα του υλικού ή του λογισμικού εκσυγχρονίζεται μ' ένα νεότερο μοντέλο ή έκδοση. Τότε το σύστημα θα πρέπει να μπορεί να μετακινηθεί στο νέο ή στο διαφορετικό μοντέλο ή έκδοση, χωρίς να καταστραφεί η λειτουργικότητά του. Γι' αυτό είναι καλό το λογισμικό να είναι **αυτοδιαιρούμενο,** δηλαδή να αποτελείται από ανεξάρτητα τμήματα *(modules),* το καθένα από τα οποία να μπορεί να διορθώνεται, αναβαθμίζεται ή συντηρείται από μόνο του.

Στη δεύτερη κατηγορία ανήκουν οι χρήστες και οι προγραμματιστές που χρησιμοποιούν το σύστημα. Το σύστημα πρέπει να είναι **αξιόπιστο** *(reliable)* και αποδοτικό. Με τον όρο αξιοπιστία εννοούμε ότι το σύστημα παράγει τα σωστά αποτελέσματα με το σωστό βαθμό ακρίβειας. Σ' αυτή την περίπτωση λέμε ότι το σύστημα διαθέτει **ακεραιότητα** *(integrity).* Επί πλέον, αν το ίδιο σύνολο δεδομένων εισόδου εισαχθεί στο σύστημα πολλές φορές, κάτω από τις ίδιες συνθήκες, τα αποτελέσματα θα είναι τα ίδια. Αυτό ονομάζεται **συνέπεια λειτουργίας** *(consistency).* Τέλος, η



Σχήμα 14.3. - Χαρακτηριστικά του ποιοτικού λογισμικού κατά Boehm

πιο χρήσιμη ιδιότητα είναι η εργονομικότητα. Οι χρήστες και οι προγραμματιστές πρέπει να βρίσκουν το σύστημα εύκολο στην εκμάθηση και τη χρήση. Ένα σύστημα μπορεί να είναι πολύ "καλό" κατά την εκτέλεση μιας λειτουργίας, αλλά οι χρήστες να μην μπορούν να αντιληφθούν πώς να το χρησιμοποιήσουν. Σε αυτή την περίπτωση το σύστημα είναι αποτυχημένο.

Στην τρίτη κατηγορία χρηστών ανήκουν οι προγραμματιστές που συντηρούν το σύστημα **(συντηρήσιμο).** Αυτοί κάνουν τις αλλαγές που χρειάζεται, κάθε φορά που οι απαιτήσεις του πελάτη αλλάζουν, ή κάθε φορά που ανιχνεύονται λάθη. Οι προγραμματιστές πρέπει να μπορούν να εντοπίσουν την αιτία ενός λάθους, να βρουν τα τμήματα (τις μονάδες του προγράμματος) που εκτελούν μια λειτουργία **(ελέγξιμο)**, να κατανοήσουν τον κώδικα **(κατανοήσιμο)** και να τον τροποποιήσουν **(τροποποιήσιμο).**

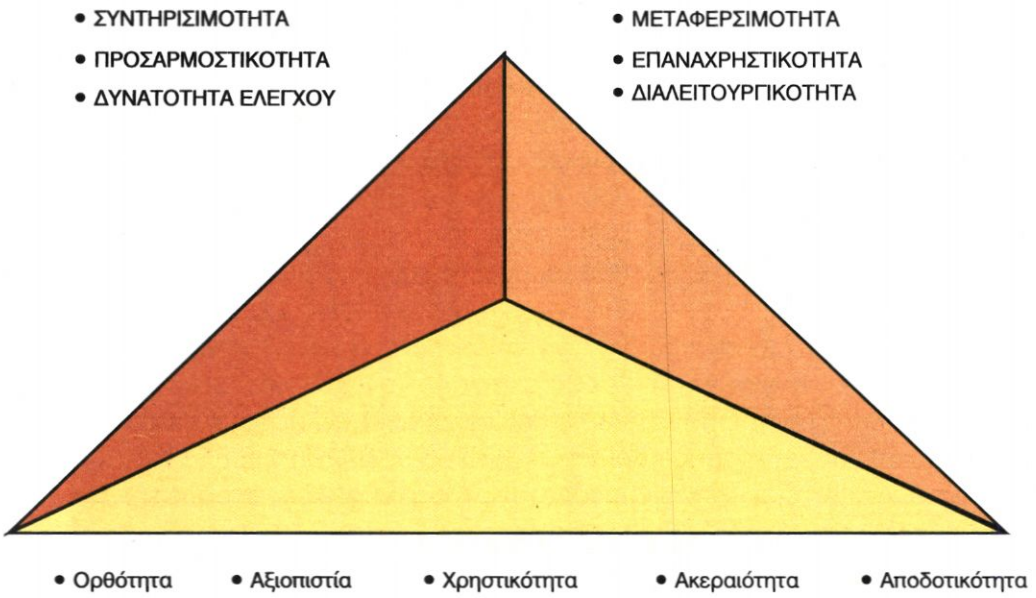
Εκτός από την προσέγγιση του Boehm, που αναφέρθηκε παραπάνω, υπάρχουν και άλλες. Μεταξύ αυτών σημαντικότερες θεωρούνται η προσέγγιση του McCall και η προσέγγιση FURPS

α) Η προσέγγιση του McCall

Ο McCall και οι συνάδελφοί του (1977) πρότειναν μια χρήσιμη κατηγοριοποίηση των παραγόντων που καθορίζουν το ποιοτικό λογισμικό. Αυτοί οι παράγοντες, όπως φαίνονται στο σχήμα 14.4, εστιάζουν σε τρεις βασικές πλευρές ενός προϊόντος λογισμικού: στα λειτουργικά του χαρακτηριστικά, στη δυνατότητά του να υφίσταται αλλαγές και στη δυνατότητα προσαρμογής σ' ένα νέο περιβάλλον.

Ο McCall δίνει τις εξής περιγραφές, όσον αφορά στους παράγοντες του σχήματος 14.4:

- Ορθότητα (correctness). Η έκταση που ένα πρόγραμμα ικανοποιεί τις προδιαγραφές και εκπληρώνει τους στόχους του πελάτη.
- Αξιοπιστία *(reliability)*. Η έκταση στην οποία ένα πρόγραμμα αναμένεται να εκτελεί τις επιδιωκόμενες λειτουργίες με την απαιτούμενη ακρίβεια. Πρέπει να σημειωθεί ότι έχουν προταθεί και άλλοι, πιο πλήρεις, ορισμοί για την αξιοπιστία (βλέπε ενότητα 14.2.4).
- Αποδοτικότητα *(efficiency)*. Το σύνολο των πόρων του υπολογιστή και του κώδικα που απαιτείται, ώστε το πρόγραμμα να εκτελεί τις λειτουργίες του.
- Ακεραιότητα *(integrity)*. Η έκταση στην οποία η προσπέλαση στο λογισμικό ή τα δεδομένα από μη εξουσιοδοτημένα πρόσωπα μπορεί να ελεγχθεί.
- Χρηστικότητα *(usability)*. Η προσπάθεια που απαιτείται για την εκμάθηση, το χειρισμό, την προετοιμασία εισαγωγής δεδομένων και την ερμηνεία των αποτελεσμάτων από ένα πρόγραμμα.
- Ευκολία συντήρησης (maintainability). Η προσπάθεια που απαιτείται για τον εντοπισμό και την επιδιόρθωση ενός σφάλματος σε ένα πρόγραμμα (ο ορισμός αυτός θεωρείται πολύ περιορισμένος).
- Προσαρμοστικότητα *(flexibility)*. Η προσπάθεια που απαιτείται για την τροποποίηση ενός προγράμματος.
- Δυνατότητα ελέγχου *(testability)*. Η προσπάθεια που απαιτείται για τον έλεγχο ενός προγράμματος, ώστε να εξασφαλιστεί ότι αυτό εκτελεί τις επιδιωκόμενες λειτουργίες.
- Μεταφερσιμότητα *(portability)*. Η προσπάθεια που απαιτείται για να μεταφερθεί το πρόγραμμα από ένα υλικό *(hardware)* και/ή από ένα λογισμικό περιβάλλον σε ένα άλλο.
- Δυνατότητα επαναχρησιμοποίησης *(reusability)*. Η έκταση στην οποία ένα πρόγραμμα (ή τμήμα του προγράμματος) μπορεί να επαναχρησιμοποιηθεί σε άλλες εφαρμογές.
- Διαλειτουργικότητα *(interoperability)*. Η προσπάθεια που απαιτείται για να συνδεθεί ένα σύστημα σε ένα άλλο.



Σχήμα 14.4. - Παράγοντες ποιότητας λογισμικού του McCall.

β) Η προσέγγιση FURPS

Η Hewlett-Packard (Grady, 1987) έχει αναπτύξει ένα σύνολο παραγόντων που προσδιορίζουν την ποιότητα του λογισμικού, στο οποίο έχει δοθεί το ακρωνύμιο FURPS - λειτουργικότητα *(Functionality)*, χρηστικότητα *(Usability)*, αξιοπιστία *(Reliability)*, επιδόσεις *(Performance)* και δυνατότητα υποστήριξης *(Supportability)*.

Ο προσδιορισμός των πέντε αυτών παραγόντων έχει ως ακολούθως:

- Λειτουργικότητα *(functionality)*. Προσδιορίζεται από τον υπολογισμό του συνόλου των χαρακτηριστικών και ικανοτήτων του προγράμματος, την έκταση των λειτουργιών και την ασφάλεια του όλου συστήματος.
- Χρηστικότητα *(usability)*. Προσδιορίζεται από θέματα που έχουν να κάνουν με τον ανθρώπινο παράγοντα, όπως αισθητική, τεκμηρίωση, συνέπεια.
- Αξιοπιστία *(reliability)*. Υπολογίζεται μετρώντας τη συχνότητα και τη σοβαρότητα βλαβών, την ακρίβεια των αποτελεσμάτων, το μέσο χρόνο μεταξύ δύο βλαβών, τη δυνατότητα επαναφοράς μετά από βλάβη και την προβλεψιμότητα του προγράμματος.
- Επιδόσεις *(performance)*. Υπολογίζεται από την ταχύτητα επεξεργασίας, τον χρόνο απόκρισης, την κατανάλωση πόρων του συστήματος και την αποδοτικότητα.
- Δυνατότητα υποστήριξης *(supportability)*. Συνδυάζει τη δυνατότητα επέκτασης των προγραμμάτων, τη δυνατότητα προσαρμογής, τη δυνατότητα υποστήριξης (τα τρία αυτά χαρακτηριστικά συνήθως αναφέρονται με τον όρο συντηρησιμότητα), καθώς και τη δυνατότητα ελέγχου, τη συμβατότητα, την ευκολία εγκατάστασης και την ευκολία με την οποία τα προβλήματα μπορούν να εντοπιστούν.

Υπάρχει ένας μεγάλος κατάλογος βιβλίων που περιγράφουν με λεπτομέρεια **τεχνικές μέτρησης**, αν και η πλειονότητα εστιάζει σε μετρήσεις διαδικασιών και έργων λογισμικού. Τέτοια βιβλία είναι των **Moeller** και **Paulish**, Software Metrics, Chapman & Hall, 1993, **Grady R. B.**, Practical Software Metrics for Project Management and Proccess Improvement, Prentice-Hall, 1992, **Ejiogu L.** Software Engineering with Formal Metrics, QED Publishing, 1991.

14.2.2. Μέτρα της ποιότητας του λογισμικού

Στις προηγούμενες ενότητες αναφέρθηκε ένα σύνολο ποιοτικών παραγόντων για τη "μέτρηση" του λογισμικού. Είναι προφανές ότι απόλυτα μέτρα όπως τάση, θερμοκρασία, ταχύτητα ή όγκος είναι ασυνήθη στον κόσμο του λογισμικού. Παρόλα αυτά έχουν γίνει πολλές προσπάθειες για την ποσοτική αξιολόγηση της ποιότητας του λογισμικού. Οι παράγοντες που καθορίζουν την ποιότητα του Λογισμικού μπορούν να χωριστούν σε δύο κατηγορίες:

1. Παράγοντες που μπορούν να μετρηθούν άμεσα (π.χ., αριθμός σφαλμάτων σε μια μονάδα προγράμματος) και
2. Παράγοντες που μπορούν να μετρηθούν έμμεσα (π.χ., χρησιμότητα ή συντηρησιμότητα).

Επειδή οι μετρήσεις του λογισμικού δεν είναι απόλυτες, είναι και ανοικτές σε διαμάχες. Πολλά μέλη της κοινωνίας του λογισμικού ισχυρίζονται ότι το λογισμικό δεν είναι μετρήσιμο ή ότι κάθε προσπάθεια μέτρησης θα πρέπει να αναβληθεί, μέχρι να καταλάβουμε καλύτερα τα χαρακτηριστικά που θα πρέπει να χρησιμοποιηθούν για την περιγραφή του. Για όλα τα παραπάνω και επειδή οι περισσότερες μετρήσεις είναι έμμεσες, στην ουσία ποτέ δε μετράμε την ποιότητα του λογισμικού, αλλά μάλλον μια εκδήλωση της ποιότητας.

Στη συνέχεια θα εξετάσουμε τρία χαρακτηριστικά που συνήθως θεωρούνται ως μέτρα της συνολικής ποιότητας του λογισμικού: την **αξιοπιστία** *(reliability)*, τη **διαθεσιμότητα** *(availability)* και τη συντηρησιμότητα *(maintainability)*. Οι όροι αυτοί είναι δανεισμένοι από το χώρο της μηχανικής, όπου τα κριτήρια αυτά χρησιμοποιούνται για να κρίνουν την ποιότητα ενός συστήματος.

Αξιοπιστία λογισμικού *(software reliability)* είναι η πιθανότητα το πρόγραμμα να λειτουργεί ικανοποιητικά για μια δεδομένη χρονική περίοδο. Αφού χρησιμοποιείται η πιθανότητα σαν μέτρο, η αξιοπιστία εκφράζεται σαν ένας αριθμός μεταξύ 0 (καθόλου αξιόπιστο) και 1 (πλήρως αξιόπιστο). Ένας άλλος απλός τρόπος μέτρησης της αξιοπιστίας είναι ο μέσος χρόνος (σε ώρες) μεταξύ δύο βλαβών (ΜΧΜΒ). Έτσι η αξιοπιστία είναι:

$$R = \frac{MXMB}{1 + MXMB}$$

Είναι προφανές ότι αν ο ΜΧΜΒ είναι μικρός, τότε το R παίρνει τιμή κοντά στο 0 (ελάχιστα αξιόπιστο σύστημα). Αντίθετα, αν ο ΜΧΜΒ αυξάνεται, τότε το R προσεγγίζει το 1 (αρκετά αξιόπιστο σύστημα).

Διαθεσιμότητα λογισμικού *(software availability)* είναι η πιθανότητα ένα πρόγραμμα να λειτουργεί ικανοποιητικά, σύμφωνα με τις απαιτήσεις του λογισμικού σε μια δεδομένη χρονική στιγμή. Η διαθεσιμότητα ορίζεται ως εξής:

$$\Delta = \frac{MXMB}{MXMB + MXE}$$

όπου ΜΧΕ=μέσος χρόνος (σε ώρες) για την επιδιόρθωση ενός σφάλματος
και ΜΧΜΒ = μέσος χρόνος (σε ώρες) μεταξύ δύο βλαβών.

"Συντηρησιμότητα" λογισμικού *(software maintainability)* είναι η ευκολία που ένα λογισμικό σφάλμα μπορεί να επιδιορθωθεί. Δεν υπάρχει τρόπος να μετρηθεί η "συντηρησιμότητα" άμεσα. Γι' αυτό πρέπει να χρησιμοποιήσουμε έμμεσες μετρήσεις. Ένας απλός τρόπος μέτρησης είναι η χρήση του μέσου χρόνου αλλαγής (ΜΧΑ). ΜΧΑ είναι ο χρόνος που απαιτείται για να αναλυθεί μια αλλαγή που απαιτείται, να σχεδιαστεί η κατάλληλη τροποποίηση, να υλοποιηθεί η αλλαγή, να ελεγχθεί και να διανεμηθεί στους χρήστες. Κατά μέσο όρο, τα προγράμματα που είναι συντηρήσιμα έχουν πιο μικρό ΜΧΑ (για ισοδύναμους τύπους αλλαγών) από τα προγράμματα που δεν είναι συντηρήσιμα.

Ένας άλλος τρόπος μέτρησης της "συντηρησιμότητας" είναι ο υπολογισμός του Δείκτη Ωριμότητας Λογισμικού (ΔΩΛ) που προτείνεται στο πρότυπο IEEE 982.1-1988. Ο ΔΩΛ παρέχει μια ένδειξη της σταθερότητας ενός λογισμικού προϊόντος (που βασίζεται στις αλλαγές που συμβαίνουν για κάθε νέα έκδοση του προϊόντος) και υπολογίζεται ως εξής:

$$\Delta\Omega\Lambda = \frac{[T - (A+B+\Gamma)]}{T}$$

όπου
Τ = ο αριθμός των μονάδων (τμημάτων) στην τρέχουσα έκδοση
Α = ο αριθμός των μονάδων στην τρέχουσα έκδοση που έχει αλλάξει
Β = ο αριθμός των μονάδων στην τρέχουσα έκδοση που έχει προστεθεί
Γ = ο αριθμός των μονάδων από τις προηγούμενες εκδόσεις που έχουν διαγραφεί στην τρέχουσα έκδοση.
Εφόσον ο ΔΩΛ προσεγγίζει το 1.0, το λογισμικό προϊόν αρχίζει να γίνεται σταθερό.

14.3. ΠΑΡΑΔΟΣΗ ΤΟΥ ΠΡΟΪΟΝΤΟΣ ΣΤΟΥΣ ΧΡΗΣΤΕΣ

Η μεταφορά της κυριότητας του συστήματος από τον κατασκευαστή του στον πελάτη δεν είναι μια απλή διαδικασία. Ο χρήστης για να είναι κύριος του συστήματος θα πρέπει αφ' ενός να το χρησιμοποιεί σωστά και αποτελεσματικά και αφ' ετέρου να είναι ευχαριστημένος από το σύστημα που παρέλαβε. Για να γίνει όμως αυτό θα πρέπει να βοηθήσουμε το χρήστη να καταλάβει και να εξοικειωθεί με το σύστημα. Με τον όρο "παράδοση του προϊόντος" εννοούμε ακριβώς τη χρονική εκείνη περίοδο, κατά την οποία προσφέρουμε αυτή τη βοήθεια στους χρήστες. Η βοήθεια αυτή μπορεί να προσφερθεί κυρίως με δύο τρόπους: την τεκμηρίωση και την εκπαίδευση.

Στην ενότητα αυτή εξετάζεται η τεκμηρίωση η οποία πρέπει να συνοδεύει το προϊόν κατά την παράδοσή του στον πελάτη, καθώς και η εκπαίδευση που πρέπει να του παρέχεται. Όσον αφορά την τεκμηρίωση, εξετάζεται :

α. η τεκμηρίωση χρήσης και χειρισμού, η οποία απευθύνεται στους χρήστες και στους χειριστές του συστήματος και έχει σκοπό τη διευκόλυνσή τους στην επίλυση οποιουδήποτε προβλήματος, ή στην άντληση κάποιας πληροφορίας και
β. η τεκμηρίωση συστήματος, που περιλαμβάνει όλα τα εγχειρίδια τεκμηρίωσης που δημιουργούνται κατά τη φάση ανάπτυξης του συστήματος, καθώς και πρόσθετα εγχειρίδια που θα βοηθήσουν όσους θα επαυξήσουν ή θα συντηρήσουν το σύστημα.

Η παράδοση του προϊόντος στον πελάτη μπορεί να θεωρηθεί πετυχημένη, αν η τεκμηρίωση και η εκπαίδευση του δίνουν τη δυνατότητα να χειρίζεται αποτελεσματικά το σύστημα.

14.3.1. Εκπαίδευση

Μια βασική διαδικασία κατά τη φάση της παράδοσης του λογισμικού προϊόντος είναι η εκπαίδευση των χρηστών και των χειριστών. Μέσω της εκπαιδευτικής διαδικασίας βοηθάμε τους χρήστες και τους χειριστές να καταλάβουν και να εξοικειωθούν με το προϊόν. Αν οι χρήστες δεν χρησιμοποιήσουν σωστά και παραγωγικά το σύστημα μετά την παράδοσή του σε αυτούς ή δεν είναι ευχαριστημένοι με την εκτέλεσή του, τότε η όλη προσπάθεια της ομάδας ανάπτυξης πηγαίνει χαμένη.

Η **εκπαίδευση των χρηστών** έχει σαν στόχο να τους παρουσιάσει τις κύριες λειτουργίες του συστήματος, ώστε αυτοί να μπορούν να καταλάβουν ποιες είναι αυτές και να τις εκτελούν. Δεν είναι απαραίτητο οι χρήστες να γνωρίζουν τους εσωτερικούς χειρισμούς του συστήματος. Η εκπαίδευση των χρηστών απαιτεί μεγάλη προσοχή, γιατί αυτοί πολλές φορές είναι αναγκασμένοι να απαρνηθούν οικείες και προσφιλείς δραστηριότητες για να μάθουν νέες.

Η **εκπαίδευση των χειριστών** εστιάζεται στις υποστηρικτικές λειτουργίες του συστήματος και στόχο έχει να τους μάθει το πως λειτουργεί το σύστημα και όχι το τι κάνει. Έτσι, ο χειριστής θα πρέπει να μάθει να παρακολουθεί και να βελτιώνει την εκτέλεση του συστήματος, να καλεί υποστηρικτικές διαδικασίες, να φροντίζει την ασφάλειά του, να παίρνει αντίγραφα ασφαλείας (backup), να επιμελείται τους χώρους των δίσκων, να φροντίζει την επικοινωνία με άλλα συστήματα κ.λ.π.

Η εκπαίδευση πρέπει να οδηγεί από το γενικό στο ειδικό και να είναι έτσι οργανωμένη, ώστε να απευθύνεται σε διαφορετικά επίπεδα χρηστών, ανάλογα με την εμπειρία, την κατανομή αρμοδιότητας στο σύστημα και την περιοδικότητα εργασίας. Είναι φανερό ότι ένας χρήστης που χρησιμοποιεί περιοδικά το σύστημα χρειάζεται διαφορετική εκπαίδευση από έναν μόνιμο χρήστη, αφού δε χρειάζεται να γνωρίζει τα ειδικά χαρακτηριστικά του συστήματος, αλλά μόνο τα βασικά. Πρέπει επίσης να έχουν σχεδιαστεί εξειδικευμένα μαθήματα για εκείνους που έχουν εξειδικευμένες απαιτήσεις. Το υλικό πάντως σε μια εκπαιδευτική τάξη θα πρέπει να διακρίνεται σε ενότητες και ο στόχος της κάθε ενότητας να οριοθετείται με σαφήνεια.

Σε κάθε περίπτωση, η εκπαιδευτική διαδικασία θα πρέπει να συνοδεύεται από εκπαιδευτικά εγχειρίδια, που θα πρέπει να έχουν δημιουργηθείί κατά τη φάση της ανάπτυξης του προϊόντος.

14.3.2. Τεκμηρίωση χρήσης και χειρισμού

Ένα σύστημα λογισμικού χρησιμοποιείται κυρίως από δύο κατηγορίες ανθρώπων: τους **χρήστες** και τους **χειριστές.** Αν δούμε το λογισμικό σαν ένα όχημα, μπορούμε να πούμε ότι ο χρήστης είναι ο οδηγός του οχήματος και ο χειριστής είναι ο μηχανικός του οχήματος. Ο μηχανικός ίσως να μην οδηγεί το όχημα, όμως το υποστηρίζει για να το οδηγεί ο οδηγός.

Ένας χρήστης εξασκεί τις κύριες λειτουργίες του λογισμικού όπως: χειρισμός αρχείων δεδομένων, εκτέλεση λειτουργιών, ανάλυση δεδομένων, επικοινωνία με το σύστημα, εξαγωγή αναφορών και γραφημάτων. Ο χειριστής ασκεί συχνά τις συμπληρωματικές λειτουργίες που υποστηρίζουν τις κύριες λειτουργίες του συστήματος όπως: προσδιορισμός των δυνατοτήτων πρόσβασης των χρηστών, εγκατάσταση νέου λογισμικού, λήψη αντιγράφων ασφαλείας (backup) κ.λ.π. Οι χρήστες, λοιπόν, και οι χειριστές πρέπει να είναι εφοδιασμένοι με εκείνα τα εγχειρίδια, τα οποία θα τους παρέχουν όλες τις πληροφορίες για την ορθή και αποτελεσματική χρήση και χειρισμό των συστημάτων:

- το εγχειρίδιο χρήσης *(user's manual)*
- το εγχειρίδιο χειρισμού *(operator's manual)*
- ο οδηγός παραπομπών για μηνύματα σφαλμάτων *(error message reference guide).*
- ο σύντομος οδηγός αναφορών *(quick reference guide)*

Το εγχειρίδιο χρήσης

Το εγχειρίδιο χρήσης *(user's manual)* είναι ο οδηγός στον οποίο μπορεί να ανατρέχει ανά πάσα στιγμή ο χρήστης του συστήματος. Στο εγχειρίδιο αυτό ο χρήστης πρέπει με εύκολο και γρήγορο τρόπο να βρίσκει τις πληροφορίες που χρειάζεται, ώστε να χρησιμοποιεί αποτελεσματικά το σύστημα. Έτσι πρέπει κατά τη συγγραφή του εγχειριδίου να λαμβάνονται υπόψη όλες οι τεχνικές αύξησης της αναγνωσιμότητας και πρόσβασης στις πληροφορίες. Απαραίτητη είναι η ύπαρξη ευρετηρίων διαγραμμάτων, λεξικών όρων, πινάκων, σχημάτων κ.λ.π.

Συνήθως το εγχειρίδιο χρήσης αρχίζει περιγράφοντας το σκοπό του, κάνοντας αναφορά και σε άλλα εγχειρίδια τεκμηρίωσης του συστήματος, που περιέχουν πιο λεπτομερείς πληροφορίες. Αυτό έχει σαν σκοπό να καθησυχάσει τους χρήστες, αποδεικνύοντας ότι το είδος πληροφοριών που αναζητούν βρίσκεται σε αυτό το έντυπο. Στη συνέχεια πολύ συνοπτικά περιγράφει:

- τις αρχές και το σκοπό του συστήματος,
- τις δυνατότητες και τις λειτουργίες του,
- τα χαρακτηριστικά και τα πλεονεκτήματά του.

Όσον αφορά στις λειτουργίες που εκτελεί το σύστημα, το εγχειρίδιο του χρήστη πρέπει να περιλαμβάνει:

- τις κύριες λειτουργίες και τον τρόπο που συνδέονται,
- περιγραφή των επιμέρους λειτουργιών (επιλογές του κάθε μενού, πλήκτρα λειτουργίας, οθόνες κ.λ.π.),
- περιγραφή των εισόδων και εξόδων κάθε λειτουργίας,
- περιγραφή των ειδικών χαρακτηριστικών κάθε λειτουργίας.

Το εγχειρίδιο χειριστή

Το εγχειρίδιο χειριστή *(operator's manual)* απευθύνεται στους χειριστές. Αυτοί ενδιαφέρονται περισσότερο για τις λεπτομέρειες εκτέλεσης των λειτουργιών του συστήματος, καθώς και για τις δυνατότητες πρόσβασης στο σύστημα. Έτσι, το εγχειρίδιο αυτό εξηγεί τις μεθόδους χορήγησης πρόσβασης στο σύστημα σ' ένα χρήστη, τις διαδικασίες λήψης αντιγράφων ασφαλείας, τις διατάξεις του λογισμικού, τεχνικές για την αντιγραφή αρχείων κ.λ.π. Πρέπει, επίσης, να παρέχει οδηγίες εγκατάστασης, απαιτήσεις χωρητικότητας και λεπτομέρειες επιδόσεων. Η δομή του εγχειριδίου χειρισμού μοιάζει με αυτό του εγχειριδίου χρηστών. Με τη βοήθεια του εγχειριδίου αυτού, οι χειριστές πρέπει να αναγνωρίζουν αν ένα πρόβλημα που τους αναφέρεται από τους χρήστες μπορεί να διορθωθεί απ' αυτούς ή από τους συντηρητές.

Οδηγός παραπομπών για μηνύματα σφαλμάτων

Σ' ένα καλά σχεδιασμένο λογισμικό, κάθε φορά που εντοπίζεται ένα σφάλμα, οι χρήστες και οι χειριστές ειδοποιούνται μ' έναν ομοιόμορφο τρόπο. Η τεκμηρίωση για το χρήστη καταγράφει όλα τα πιθανά σφάλματα και τη σημασία τους, υποδεικνύοντας πολλές φορές και την πηγή του σφάλματος. Ένα μήνυμα σφάλματος, όταν εντοπίζεται, εμφανίζεται στη οθόνη περιλαμβάνοντας συνήθως τις εξής πληροφορίες:

- κωδικός σφάλματος,
- μονάδα του προγράμματος που εκτελείται κατά τη διάρκεια εμφάνισης του σφάλματος,
- τη σοβαρότητα του σφάλματος και τις συνέπειες στο σύστημα,
- τα περιεχόμενα κάθε σχετικής μνήμης του συστήματος.

Πολλές φορές, όμως, δεν εμφανίζεται στην οθόνη η αιτία του σφάλματος, ή δεν υπάρχει αρκετός χώρος ώστε να παρουσιαστεί το πλήρες μήνυμα. Για το λόγο αυτό πρέπει να είναι διαθέσιμος **ο οδηγός παραπομπών για μηνύματα σφαλμάτων** *(error message reference guide)*, που θα περιγράφει πλήρως το σφάλμα. Σ' αυτό θα ανατρέχει ο χειριστής ή ο χρήστης, κάθε φορά που βλέπει κάποιο κωδικό σφάλματος.

Σύντομος οδηγός αναφορών

Ο **σύντομος οδηγός αναφορών** *(quick reference guide)* περιέχει σε λίγες σελίδες (ίσως μία ή δύο πολλές φορές) μια περίληψη των βασικών λειτουργιών του συστήματος. Αυτός ο οδηγός είναι πολύ χρήσιμος στους χρήστες ή στους χειριστές αφού, χωρίς μακροσκελείς εξηγήσεις της μεθόδου εκτέλεσης μιας λειτουργίας, μπορούν να βρουν τον τρόπο εκτέλεσής της. Αυτός ο οδηγός, πολλές φορές περιέχει ειδικούς προσδιορισμούς πλήκτρων και κωδικούς ή συντομογραφίες. Σε πιο εξελιγμένα λογισμικά συστήματα, ο οδηγός αναφορών μπορεί να εμφανίζεται στη οθόνη με το πάτημα ενός πλήκτρου.

14.3.3. Τεκμηρίωση συστήματος

Εκτός από την τεκμηρίωση χρήσης και χειρισμού, πολλά ακόμα έγγραφα τεκμηρίωσης μπορούν να συνοδεύουν το σύστημα κατά την παράδοσή του στους πελάτες. Άλλα από αυτά έχουν δημιουργηθεί στη φάση της ανάπτυξης του λογισμικού (τεκμηρίωση απαιτήσεων, τεκμηρίωση σχε-

δίου, τεκμηρίωση προγραμμάτων, τεκμηρίωση ελέγχου) και άλλα δημιουργούνται με σκοπό να βοηθήσουν αυτούς που θα επαυξήσουν ή θα συντηρήσουν το σύστημα (οδηγός προγραμματιστή και οδηγός συντήρησης).

Ο οδηγός συντήρησης *(maintenance guide)* έχει στόχο να βοηθήσει τους συντηρητές του συστήματος στον εντοπισμό των προβλημάτων. Περιγράφει λειτουργίες υποστήριξης του συστήματος (εκτέλεση διαγνωστικών προγραμμάτων, σημεία του κώδικα που πρέπει να ελεγχθούν), ενώ παρέχει βοήθεια στην περίπτωση επαύξησης του συστήματος. Αν, για παράδειγμα, προστεθεί ένα νέο υλικό στη διάταξη *(configuration)* του συστήματος, ο οδηγός συντήρησης παρέχει τα διαθέσιμα εργαλεία για την ενημέρωση του κώδικα.

Τέλος, ο **οδηγός προγραμματιστή** *(programmer's guide)* είναι ένα εγχειρίδιο που περιγράφει μια συνολική θεώρηση του τρόπου διάταξης του υλικού και λογισμικού, καθώς και αναλυτική περιγραφή των επιμέρους μονάδων του προγράμματος και του τρόπου επικοινωνίας μεταξύ τους. Το εγχειρίδιο αυτό μπορεί να βοηθήσει τον προγραμματιστή να εντοπίσει ακριβώς τον κώδικα από τον οποίο προήλθε ένα σφάλμα ή το σημείο του κώδικα που πρέπει να τροποποιηθεί προκειμένου να γίνει κάποια επαύξηση στη διάταξη του υλικού ή να τροποποιηθεί μια λειτουργία.

Ανακεφαλαίωση / περίληψη

Με τον όρο "τεκμηρίωση λογισμικού" εννοούμε το σύνολο των πληροφοριών που έχουν σχεδιαστεί με σκοπό να διευκολύνουν την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν. Η τεκμηρίωση είναι βασικό συστατικό στην κατασκευή οποιουδήποτε λογισμικού και δε συνιστά μία ξεχωριστή φάση στον κύκλο ζωής ενός συστήματος. Αντίθετα, δημιουργείται σταδιακά κατά τη διάρκεια των διαφόρων φάσεων ανάπτυξης του συστήματος.

Η τεκμηρίωση μπορεί να ταξινομηθεί με βάση το περιβάλλον στο οποίο χρησιμοποιείται (εσωτερική και εξωτερική τεκμηρίωση), με βάση το ακροατήριο στο οποίο απευθύνεται (τεκμηρίωση για τη διοίκηση, για το χρήστη, για τον προγραμματιστή, για τη λειτουργία) ή με βάση τη φάση ανάπτυξης του λογισμικού (τεκμηρίωση απαιτήσεων, σχεδίου, προγράμματος, ελέγχου, χρήσης και χειρισμού, συστήματος). Για την πληρέστερη σύνταξη των εγγράφων τεκμηρίωσης έχουν αναπτυχθεί από διάφορους οργανισμούς (όπως ISO, IEEE) κάποια πρότυπα τεκμηρίωσης που σκοπό έχουν να αποτελέσουν οδηγούς για τη συγγραφή κάθε εγγράφου τεκμηρίωσης καθορίζοντας το σκοπό, το περίγραμμα, το περιεχόμενο καθώς και τα ποιοτικά του χαρακτηριστικά.

Τα τελευταία χρόνια έχει αυξηθεί σε μεγάλο βαθμό η χρησιμοποίηση της τεκμηρίωσης άμεσης επικοινωνίας (on-line documentation), η οποία ξεπερνά τα όρια των παραδοσιακών εγχειριδίων, στο μέτρο που έχει δυνατότητα να ενσωματώνει κείμενο, γραφικά, κίνηση, ηχητικά εφφέ κ.λ.π. Η ανάπτυξη των δικτύων και η δυνατότητα ηλεκτρονικής δημοσίευσης έχει συντελέσει σε μεγάλο βαθμό σ' αυτή την ανάπτυξη.

Η αξιολόγηση του λογισμικού γίνεται με βάση την έμμεση ή άμεση μέτρηση παραγόντων ή χαρακτηριστικών που καθορίζουν την ποιότητα του λογισμικού. Στη βιβλιογραφία έχουν καταγραφεί διάφορες προσεγγίσεις (του Boehm, του McCall, FURPS), οι οποίες προτείνουν διάφορες κατηγορίες παραγόντων που προσδιορίζουν την ποιότητα του λογισμικού όπως είναι η ορθότητα, η αξιοπιστία, η αποδοτικότητα, η ακεραιότητα, η προσαρμοστικότητα, η μεταφερτότητα, η λειτουργικότητα κ.α.

Η μεταφορά της κυριότητας μιας εφαρμογής λογισμικού από τον κατασκευαστή στο χρήστη (ή τον πελάτη) θα πρέπει να συνοδεύεται από ένα πλήρες σύνολο εγχειριδίων, που παρέχουν οδηγίες χρήσης, λειτουργίας και εκμετάλλευσης της εφαρμογής, σε όλους όσους πρόκειται να έρθουν σε επαφή με αυτή. Τέλος, μεγάλη σημασία πρέπει να δίνεται στην εκπαίδευση των χρηστών και χειριστών στην εφαρμογή, πριν αυτή τεθεί σε πραγματική λειτουργία.

Ερωτήσεις - Δραστηριότητες -Θέματα προς συζήτηση

1. Συμπληρώστε τις παρακάτω φράσεις:

α) Ο μέσος χρόνος αλλαγής (ΜΧΑ) είναι ένας απλός τρόπος μέτρησης της .....

β) ..... είναι η πιθανότητα το πρόγραμμα να λειτουργεί ικανοποιητικά για μια δεδομένη χρονική περίοδο.

γ) ..... είναι η πιθανότητα ένα πρόγραμμα να λειτουργεί ικανοποιητικά σύμφωνα με τις απαιτήσεις του λογισμικού σε μια δεδομένη χρονική στιγμή.

δ) Ο δείκτης ωριμότητας λογισμικού (ΔΟΛ) είναι ένας τρόπος μέτρησης της .....

ε) Ο ..... περιέχει σε λίγες σελίδες μια περίληψη των βασικών λειτουργιών του συστήματος.

2. Η προσέγγιση του McCall εστιάζει στις τρεις κατηγορίες παραγόντων της δεξιάς στήλης. Στην κάθε κατηγορία ανήκουν παράγοντες της αριστερής στήλης. Κάνετε τη σωστή αντιστοίχηση.

Διαλειτουργικότητα	•	
Αποδοτικότητα	•	
Χρηστικότητα	•	• Λειτουργικά χαρακτηριστικά
Ευκολία συντήρησης	•	
Αξιοπιστία	•	• Προσαρμογή στο νέο περιβάλλον
Μεταφερσιμότητα	•	
Δυνατότητα ελέγχου	•	• Δυνατότητα αλλαγών
Ακεραιότητα	•	
Προσαρμοστικότητα	•	
Δυνατότητα επαναχρησιμοποίησης	•	
Ορθότητα	•	

3. Απαντήστε με σωστό ή λάθος:

α) Η προσέγγιση του McCall προτείνει μια κατηγοριοποίηση παραγόντων που καθορίζουν την αξιοπιστία λογισμικού.

β) Το προκαταρκτικό εγχειρίδιο χρήσης ανήκει στην τεκμηρίωση απαιτήσεων.

γ) Ο μέσος χρόνος μεταξύ βλαβών είναι ένας τρόπος μέτρησης της αξιοπιστίας.

δ)Ο μέσος χρόνος αλλαγής (ΜΧΑ) είναι ένας τρόπος μέτρησης της διαθεσιμότητας.

ε) Ο Δείκτης Ωριμότητας Λογισμικού παρέχει μία ένδειξη της σταθερότητας του λογισμικού και βασίζεται στις αλλαγές που συμβαίνουν σε κάθε νέα έκδοση του προϊόντος.

4. Προσπαθήστε να αιτιολογήσετε γιατί το κάθε χαρακτηριστικό, όπως εμφανίζεται στο σχήμα 14.3 (χαρακτηριστικά του ποιοτικού λογισμικού κατά Boehm), συμβάλλει στη ποιότητα του λογισμικού.

Γλωσσάριο - Ευρετήριο όρων

αξιοπιστία λογισμικού *(software reliability)* - Η ικανότητα ένα πρόγραμμα να λειτουργεί ικανοποιητικά κάτω από ορισμένες συνθήκες και για μια δεδομένη χρονική περίοδο [266].

εγχειρίδιο χειριστή *(operator's manual)* - Εγχειρίδιο τεκμηρίωσης, στο οποίο ανατρέχει ο χειριστής και περιέχει οδηγίες εγκατάστασης, απαιτήσεις αποθήκευσης, λεπτομέρειες επιδόσεων [270].

εγχειρίδιο χρήσης *(user's manual)* - Εγχειρίδιο τεκμηρίωσης, στο οποίο ανατρέχει ο χρήστης για να βρίσκει τις απαραίτητες πληροφορίες ώστε να χρησιμοποιεί αποτελεσματικά το σύστημα [269].

οδηγός παραπομπών για μηνύματα σφαλμάτων *(error message reference guide)*- Εγχειρίδιο τεκμηρίωσης που περιγράφει πλήρως τα σφάλματα του συστήματος [270].

οδηγός συντήρησης *(maintenance guide)* - Εγχειρίδιο τεκμηρίωσης, που σκοπό έχει να παρέχει βοήθεια στη συντήρηση, υποστήριξη και επαύξηση του συστήματος [271].

σύντομος οδηγός αναφορών *(quick reference guide)* - Εγχειρίδιο τεκμηρίωσης, που περιέχει μια περίληψη των βασικών λειτουργιών του συστήματος [270].

τεκμηρίωση λογισμικού *(software documentation)* - Η πληροφορία που έχει σχεδιαστεί με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν [255].

τεκμηρίωση άμεσης επικοινωνίας *(on-line documentation)* - Η πληροφορία που έχει σχεδιαστεί να παρουσιάζεται από την οθόνη, με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν [259].

Βιβλιογραφία

Brockmann R. John, "Writing Better Computer User Documentation: From Paper to Hypertext", John Wiley & Sons, 1990.

Grady R. B. and Caswell D. L., "Software Metrics: Establishing a Company-wide Program", Prentice-Hall, 1987.

Pressman S Rogers, "Software Engineering, Fourth Edition", McGraw-Hill Companies Inc., 1997.

Simpson H. and Casey S., "Developing Effective User Documentation: A Human-Factors approach", McGraw-Hill Book Company, New York, 1998.

Γιακουμάκης Α. Εμμ., "Τεχνολογία Λογισμικού", Τόμος Α, Εκδόσεις Α.Σταμούλης, 1994.

ΓΛΩΣΣΑΡΙΟ - ΕΥΡΕΤΗΡΙΟ ΟΡΩΝ

Α

αλγόριθμος *(algorithm)* - Μια προσδιορισμένη με ακρίβεια ακολουθία εντολών για την επίλυση ενός προβλήματος.

ανάλυση συστήματος ή συστημάτων *(system ή systems analysis)* - Επιστημονική μέθοδος μελέτης του ρόλου ενός προτεινόμενου συστήματος με κύριο σκοπό τον προσδιορισμό του συνόλου των απαιτήσεων, τις οποίες πρέπει να ικανοποιεί [78].

αναμενόμενα αποτελέσματα *(expected results)* - Αποτελέσματα με βάση τα οποία είναι δυνατό να ελεγχθεί, εάν η υλοποίηση των λειτουργιών έχει πραγματοποιηθεί σωστά [218].

αντικειμενοστρεφείς βάσεις δεδομένων *(object oriented DBMS)* - Βάσεις δεδομένων που περιέχουν όχι μόνο τα δεδομένα και την περιγραφή της δομής του κάθε αρχείου, αλλά και τις μεθόδους και συναρτήσεις επεξεργασίας της καταχωρημένης πληροφορίας. Χρησιμοποιούνται σε μικρή έως σήμερα κλίμακα, κυρίως σε εφαρμογές σχεδίασης (CAD) και παραγωγής (CAM) με Η/Υ, σε διαχείριση πολυμέσων και γενικά σε διαχείριση σύνθετων δεδομένων [63].

αξιοπιστία λογισμικού *(software reliability)* - Η ικανότητα ένα πρόγραμμα να λειτουργεί ικανοποιητικά κάτω από ορισμένες συνθήκες και για μια δεδομένη χρονική περίοδο [266].

απαλοιφή αιτίου *(cause elimination)* - Μέθοδος εκσφαλμάτωσης που εξετάζει τα δεδομένα που σχετίζονται με τα σφάλματα που ανιχνεύτηκαν και τα ταξινομεί σε κατηγορίες [229].

αποτύπωμα μνήμης *(memory dump)* - Τεχνική που επιτρέπει την εκτύπωση των τιμών των μεταβλητών κατά τη διάρκεια εκτέλεσης του προγράμματος [230].

αρχεία δεικτών *(index files)* - Βοηθητικά αρχεία που λειτουργούν σαν ευρετήρια, τα οποία περιέχουν κλειδιά και διευθύνσεις, που αντιστοιχούν σε εγγραφές ενός βασικού αρχείου, το οποίο δεν είναι κατ' ανάγκη ταξινομημένο. Τα αρχεία δεικτών αυξάνουν την ταχύτητα αναζήτησης και εντοπισμού δεδομένων στο αρχείο [119]

αρχεία τύπου ASCII - Αρχεία τα οποία αποτελούνται μόνο από εκτυπώσιμους χαρακτήρες του κώδικα ASCII και είναι αναγνώσιμα από οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου [179].

αρχείο *(file)* - Ένας χώρος αποθήκευσης δεδομένων ή εγγραφών που λαμβάνεται σαν ενιαία και αυτοτελής ομάδα [152].

αφορμή εκκίνησης *(trigger)* - Ένα γεγονός - συμβάν που προκαλεί την άμεση εκτέλεση ενός προγράμματος [151].

Β

βάση δεδομένων - Μια συλλογή δεδομένων οργανωμένη με σαφήνεια [25].

βιβλιοθήκη λογισμικού *(software librarie)* - Συλλογή προγραμμάτων που που χρησιμοποιούνται συχνα [198].

Γ

γεννήτρια αναφορών *(report generator)* - Πρόγραμμα που παράγει με εύκολο και γρήγορο τρόπο εκτυπώσιμες αναφορές, οι οποίες μπορούν να χρησιμοποιηθούν ανεξάρτητα ή να ενσωματωθούν σε ολοκληρωμένες εφαρμογές [124].

Δ

διάγραμμα δομής *(structure chart)* - Ένα ιεραρχικό διάγραμμα που παρουσιάζει τη συνολική αρχιτεκτονική δομή ενός προγράμματος [154].

διαγράμματα HIPO *(Hierarchy and Input - Process - Output)* - Ειδική κατηγορία διαγραμμάτων που χρησιμοποιούνται στη φάση της ανάλυσης ενός συστήματος για να παρουσιάσουν τις βασικές λειτουργίες ενός συστήματος και τον τρόπο σύνδεσής τους σε ιεραρχικά δένδρα [107].

διαγράμματα οντοτήτων - συσχετίσεων *(entity - relationship diagram)* - Ειδικά διαγράμματα τα οποία απεικονίζουν με γλαφυρότητα τις οντότητες που θα χρησιμοποιηθούν από μία εφαρμογή, καθώς και τον τρόπο με τον οποίο συσχετίζονται [130].

διαγράμματα ροής δεδομένων *(data flow diagrams)* - Τα διαγράμματα αυτά γνωστά και από τη συντομογραφία τους σαν ΔΡΔ, απεικονίζουν με ειδικά σύμβολα, τη ροή των δεδομένων μεταξύ των διαδικασιών ενός συστήματος, υποβοηθώντας σε μεγάλο βαθμό την κατανόηση των εννοιών που εμπλέκονται στο σύστημα [100].

διαδικασία *(procedure)* - Ακριβής και λεπτομερής ακολουθία γραπτών ή άγραφων οδηγιών, που βήμα προς βήμα εξηγεί τι πρέπει να γίνει, ποιος θα το κάνει, πότε θα γίνει και πως θα γίνει [74]

διαδίκτυο *(internet)* - Το παγκόσμιο δίκτυο των δικτύων [71]

διαχείριση διάταξης λογισμικού *(software configuration management)* - Η διαχείριση και ο έλεγχος όλων των τύπων τροποποιήσεων που μπορεί να υποστεί ένα σύστημα, ώστε να είναι πάντα γνωστή η κατάσταση κάθε συστατικού μέρους του [232].

δίκτυα ευρείας περιοχής (WAN) *(Wide Area Networks)* - Συστήματα υπολογιστών διασυνδεδεμένα μεταξύ τους σε δίκτυα, τα οποία είναι εγκατεστημένα σε διαφορετικούς, απομακρυσμένους φυσικούς χώρους και χρησιμοποιούν δημόσιο μέσο σύνδεσης (π.χ., γραμμές ΟΤΕ) [178].

διορθωτές *(debuggers)* - Εργαλεία λογισμικού που χρησιμοποιούνται στη διαδικασία της εκσφαλμάτωσης [229].

δοκιμαστικά δεδομένα *(test data)* - Δεδομένα που επιλέγονται με σκοπό τη δοκιμή ενός προγράμματος [217].

δοκιμή "άλφα" *(alpha test)* - Μια μορφή ελέγχου αποδοχής, που διενεργείται από τον πελάτη στους χώρους του κατασκευαστή του λογισμικού [246].

δοκιμή "βήτα" *(beta test)* - Μια μορφή ελέγχου αποδοχής που διενεργείται από τους τελικούς χρήστες στο χώρο του πελάτη [246].

δομές δεδομένων *(data structures)* - Είναι σύνολα δεδομένων μαζί με το σύνολο των επιτρεπτών σε αυτά πράξεων και λειτουργιών [198].

δομημένη γλώσσα ερωτήσεων *(Structured Query Language - SQL)* - Πρότυπη γλώσσα για την επικοινωνία του χρήστη και του προγραμματιστή με όλα τα σχεσιακά ΣΔΒΔ [31].

δομημένος προγραμματισμός *(structured programming)* - Προγραμματισμός κατά τον οποίο, αρχικά ορίζεται/σχεδιάζεται συνολικά το πρόγραμμα, στη συνέχεια οι μονάδες, οι ενότητες, οι διαδικασίες και οι συναρτήσεις και στο τέλος υλοποιούνται σε πηγαίο κώδικα οι λεπτομέρειες του [195].

E

εγχειρίδιο χειριστή *(operator's manual)* - Εγχειρίδιο τεκμηρίωσης, στο οποίο ανατρέχει ο χειριστής και περιέχει οδηγίες εγκατάστασης, απαιτήσεις αποθήκευσης, λεπτομέρειες επιδόσεων [270].

εγχειρίδιο χρήσης *(user's manual)* - Εγχειρίδιο τεκμηρίωσης, στο οποίο ανατρέχει ο χρήστης για να βρίσκει τις απαραίτητες πληροφορίες ώστε να χρησιμοποιεί αποτελεσματικά το σύστημα [269].

είσοδος(οι) *(input)* - Ορισμένη επίδραση του περιβάλλοντος στο σύστημα (ότι μπαίνει μέσα στο σύστημα και επηρεάζει τη συμπεριφορά του. Συχνά οι είσοδοι αναφέρονται και σαν εισερχόμενα δεδομένα, στοιχεία εισόδου ή εισροές [77]

εκσφαλμάτωση *(debugging)* - Διαδικασία που αποσκοπεί στον εντοπισμό και τη διόρθωση των σφαλμάτων του κώδικα ενός προγράμματος [225].

εκσφαλμάτωση *(debugging)* – Διαδικασία που αποσκοπεί στην ανίχνευση, τον εντοπισμό και τη διόρθωση των σφαλμάτων του κώδικα του λογισμικού [217].

έλεγχος απόδοσης *(benchmarking test)* - Μια μορφή ελέγχου αποδοχής που περιλαμβάνει ειδικές δοκιμές και διενεργείται από τους πραγματικούς χρήστες [246].

έλεγχος αποδοχής *(acceptance testing)* - Στάδιο του τελικού ελέγχου, που συγκρίνει το λογισμικό με τις αρχικές απαιτήσεις και τις τρέχουσες ανάγκες του τελικού χρήστη [246].

έλεγχος εγκατάστασης *(installation testing)* - Στάδιο του τελικού ελέγχου, που εντοπίζει τα σφάλματα τα οποία γίνονται κατά τη διάρκεια της εγκατάστασης του λογισμικού [247].

έλεγχος ενοτήτων *(module testing)* - Η φάση αυτή περιλαμβάνει τον ανεξάρτητο έλεγχο του συνόλου των ενοτήτων που υπάρχουν στα υποσυστήματα. Τα υποσυστήματα μπορεί να έχουν σχεδιαστεί και υλοποιηθεί ανεξάρτητα και τα πιο συνήθη προβλήματα στα μεγάλα συστήματα λογισμικού προκύπτουν στο στάδιο αυτό και αφορούν στον τρόπο διεπαφής των μονάδων των υποσυστημάτων [210].

έλεγχος λειτουργίας *(function testing)* - Στάδιο του τελικού ελέγχου, που εξετάζει αν το ολοκληρωμένο σύστημα πληροί τις λειτουργικές του απαιτήσεις [244].

έλεγχος λογισμικού *(software testing)* - Η διαδικασία της εκτέλεσης ενός προγράμματος με δοκιμαστικά δεδομένα και σύγκριση των αποτελεσμάτων του με αποτελέσματα υπολογισμών, που έγιναν με το χέρι ή με τα αποτελέσματα μιας άλλης δοκιμασμένης μεθόδου [209].

έλεγχος οριακών περιπτώσεων δεδομένων *(boundary conditions testing)* - Τύπος ελέγχου που αποκαλύπτει τη συμπεριφορά του προγράμματος όταν διοχετευτούν σε αυτό δεδομένα εισόδου τα οποία βρίσκονται στα όρια του διαστήματος του πεδίου ορισμού των δεδομένων εισόδου [221].

έλεγχος παλινδρόμησης *(regression testing)* -

Δραστηριότητα που βοηθάει στο να εξασφαλιστεί ότι οι αλλαγές που γίνονται στον κώδικα ενός προγράμματος δε θα δημιουργήσουν απρόσμενη συμπεριφορά ή επιπρόσθετα σφάλματα [231].

έλεγχος συστήματος *(system testing)* - Στάδιο του τελικού ελέγχου, που συγκρίνει το ολοκληρωμένο σύστημα με τις μη λειτουργικές του απαιτήσεις [244].

έλεγχος υποσυστημάτων *(subsystem testing)* - Οι προγραμματιστικές ενότητες που συνιστούν ένα υποσύστημα ελέγχονται σε συνεργασία μεταξύ τους, προκειμένου να ελεγχθεί η ανεξάρτητη λειτουργία των υποσυστημάτων [210].

έξοδος(οι) *(output)* - Ορισμένη επίδραση του συστήματος στο περιβάλλον - δηλαδή ορισμένη συμπεριφορά του συστήματος που επηρεάζει το περιβάλλον. Συχνά οι έξοδοι αναφέρονται και σαν εξερχόμενα δεδομένα, στοιχεία εξόδου ή εκροές [77]

εξυπηρετητής ιστού *(web server)* - Λογισμικό που δέχεται αιτήσεις από τους φυλλομετρητές ιστοσελίδων και αναλαμβάνει να τους τροφοδοτήσει με ιστοσελίδες ή άλλα δεδομένα (π.χ., αρχεία για μεταφορά). Το πρωτόκολλο επικοινωνίας του εξυπηρετητή ιστού με τους φυλλομετρητές, ονομάζεται HTTP [58].

επαλήθευση και επικύρωση *(verification & validation)* - Γενικός όρος για δοκιμές που πιστοποιούν ότι οι απαιτήσεις και οι στόχοι ενός συστήματος πληρούνται. Η επαλήθευση αξιολογεί με ποσοτικό τρόπο πόσο καλά οι απαιτήσεις ικανοποιούν τους στόχους του συστήματος [208].

επεξεργασία *(process)* - Τμήμα του συστήματος, το οποίο μετασχηματίζει δεδομένα εισόδου (inputs) σε δεδομένα εξόδου *(outputs)* [151].

εργαλεία βοηθούμενης από υπολογιστή σχεδίασης λογισμικού *(CASE Tools) (Computer Aided Software Engineering)* - Προγράμματα τα οποία βοηθούν στο σχεδιασμό και την υλοποίηση μιας εφαρμογής, ξεκινώντας από τα πρώτα στάδια και καταλήγοντας στην παραγωγή του μεγαλύτερου μέρους του κώδικά της [135].

H

HTML *(Hyper Text Markup Language)* - Γλώσσα περιγραφής και μορφοποίησης εγγράφων. Σε αυτήν συντάσσονται σήμερα οι ιστοσελίδες του Παγκόσμιου Ιστού. Βασίζεται στη χρήση ετικετών *(tags)* με τις οποίες ορίζονται παράγραφοι, μορφοποιούνται επικεφαλίδες, σχηματίζονται λίστες κ.λ.π. [57].

I

ιστοσελίδες *(web pages)* - Σελίδες υπερκειμένου που δημοσιεύονται στο διαδίκτυο [73]

ιχνηλάτηση ροής *(flow trace)* - Τεχνική κατά την οποία καθορίζονται δύο βασικά σημεία στον κώδικα ενός προγράμματος και ζητείται η εκτέλεση του προγράμματος μεταξύ αυτών των δύο σημείων [230].

J

Java - Αντικειμενοστρεφής γλώσσα προγραμματισμού κατασκευασμένη από την εταιρία Sun. Μπορεί να κατασκευάσει τόσο ολοκληρωμένες εφαρμογές, όσο και μικρές εφαρμογές *(java applets)*, που ενσωματώνονται μέσα σε ιστοσελίδες. Το μεγάλο της πλεονέκτημα είναι ότι οι εφαρμογές που κατασκευάζονται με αυτήν

μπορούν να εκτελεστούν σε μια πληθώρα υπολογιστών και λειτουργικών συστημάτων (Unix, Windows, MacOS, OS/2 κ.λ.π.) [59].
Javascript – Γλώσσα προγραμματισμού οι εφαρμογές της οποίας εκτελούνται μέσω διερμηνέα (*interpreter*). Μπορεί να ενσωματωθεί σε ιστοσελίδες και να διερμηνευθεί από το φυλλομετρητή, ή να διερμηνευθεί από τον εξυπηρετητή ιστού [59].
JDBC *(Java Data Base Modeling Language)* – Πρωτόκολλο για σύνδεση εφαρμογών Java με βάσεις δεδομένων [60].

Κ

κανονικοποίηση *(normalization)* - Διαδικασία με την οποία οργανώνονται οι οντότητες μιας εφαρμογής, με βάση τη σχεσιακή θεωρία και τους κανόνες της [130].
κατανεμημένες βάσεις δεδομένων *(distributed databases)* - Βάσεις δεδομένων, οι οποίες με τη βοήθεια της τεχνολογίας των σύγχρονων δικτύων, μοιράζουν τον όγκο των πληροφοριών τους σε διαφορετικά υπολογιστικά συστήματα, χωρίς οι χρήστες τους να αντιλαμβάνονται πού ακριβώς φυλάσσεται κάθε πληροφορία [126].
κεντρικός μετασχηματισμός - Ένα σύνολο επεξεργασιών που εκτελεί επεξεργασία δεδομένων εισόδου και παράγει δεδομένα εξόδου [163].
κοινή διεπαφή πύλης *(Common Gateway Interface CGI)* – Ένας πρότυπος τρόπος, με τον οποίο ένας εξυπηρετητής του Παγκόσμιου Ιστού περνά το αίτημα του χρήστη σε μια εφαρμογή, αποδέχεται το αποτέλεσμα της επεξεργασίας και το επιστρέφει μέσω του Ιστού στο χρήστη [60].

Λ

λειτουργικές απαιτήσεις - Περιγράφουν τι είδους επαφή πρέπει να υπάρχει ανάμεσα στο σύστημα και το περιβάλλον του. Δηλαδή, πώς το σύστημα πρέπει να αποκρίνεται όταν δέχεται συγκεκριμένα δεδομένα και τι είδους αποτελέσματα πρέπει να παράγει ανάλογα με τα εισερχόμενα αιτήματα [96].
λεξικό δεδομένων *(data dictionary)* - Ευρετήριο στο οποίο επεξηγούνται όλοι οι όροι που χρησιμοποιούνται στη φάση της ανάλυσης και του σχεδιασμού και αναφέρονται σε διεργασίες, ροές, αποθήκες δεδομένων αλλά και στα ίδια τα δεδομένα [106].
λογισμικό *(software)* - Οι αλγόριθμοι, οι δομές δεδομένων και τα κείμενα που περιγράφουν τη χρήση και τις λειτουργίες των προγραμμάτων [19].

Μ

μεταθανάτιο αποτύπωμα *(post-mortem dump)* - Τεχνική κατά την οποία εκτυπώνονται οι τιμές όλων ή επιλεγμένων μεταβλητών, σε περίπτωση βίαιου τερματισμού της εκτέλεσης του προγράμματος [230].
μοντέλο δεδομένων *(data model)* - Το είδος των δομών δεδομένων που χρησιμοποιεί το κάθε ΣΔΒΔ. Με βάση αυτό, τα ΣΔΒΔ χωρίζονται σε ιεραρχικά, δικτυωτά και σχεσιακά [30].

Ξ

ξένο κλειδί *(foreign key)* - Μια στήλη ή συνδυασμός στηλών ενός πίνακα, που αντιστοιχούν σε στήλες ενός άλλου πίνακα, δημιουργώντας το συσχετισμό μεταξύ αυτών των πινάκων [128].

Ο

οδηγοί σύνδεσης αντικειμένων με βάσεις δεδομένων - Βιβλιοθήκες κώδικα που επιτρέπουν τη σύνδεση διαφορετικών προγραμμάτων, τα οποία λειτουργούν σε περιβάλλον MS-Windows, με όλες τις δημοφιλείς βάσεις δεδομένων, έτσι ώστε να εξασφαλίζουν τη μέγιστη δυνατή μεταφερσιμότητα μεταξύ των εφαρμογών και την ανεξαρτησία τους από την επιλογή ΣΔΒΔ [179].

οδηγός παραπομπών για μηνύματα σφαλμάτων *(error message reference guide)*- Εγχειρίδιο τεκμηρίωσης που περιγράφει πλήρως τα σφάλματα του συστήματος [270].

οδηγός συντήρησης *(maintenance guide)* - Εγχειρίδιο τεκμηρίωσης, που σκοπό έχει να παρέχει βοήθεια στη συντήρηση, υποστήριξη και επαύξηση του συστήματος [271].

ολοκληρωμένη σύνθεση "από κάτω προς τα πάνω" *(integrated bottom up development)* - Μέθοδος σχεδίασης των συνιστωσών του λογισμικού κατά την οποία πρώτα ορίζονται τα χαμηλότερα επίπεδα, στη συνέχεια τα αμέσως ανώτερα από αυτά κ.ο.κ. μέχρι τον ορισμό του ανώτατου ιεραρχικού επιπέδου, την πλήρη δηλαδή περιγραφή της εφαρμογής λογισμικού [205].

οντότητα *(entity)* - Κάθε συστατικό ενός υπό εξέταση συστήματος, για το οποίο μας ενδιαφέρει να συλλέξουμε πληροφορίες Σε μία βάση δεδομένων αναφέρεται στο είδος της πληροφορίας που αποθηκεύεται, π.χ ένα πρόσωπο, μία έννοια, ένα γεγονός. Η οντότητα όταν χρησιμοποιείται κατά το σχεδιασμό μίας βάσης δεδομένων, αποτελεί τον πρόγονο ενός πίνακα [29].

οντότητα *(entity)* - Κάθε συστατικό ενός υπό εξέταση συστήματος, για το οποίο μας ενδιαφέρει να συλλέξουμε πληροφορίες. Σε μία βάση δεδομένων αναφέρεται στο είδος της πληροφορίας που αποθηκεύεται, π.χ., ένα πρόσωπο, μία έννοια, ένα γεγονός Η οντότητα όταν χρησιμοποιείται κατά το σχεδιασμό μίας βάσης δεδομένων, αποτελεί τον πρόγονο ενός πίνακα [126].

Π

παράλληλος έλεγχος *(parallel testing)* - Μια μορφή ελέγχου αποδοχής που διενεργείται στην περίπτωση που ένα νέο σύστημα αντικαθιστά ένα υπάρχον [247].

πίνακας *(table)* - Στις βάσεις δεδομένων οι πληροφορίες δεν οργανώνονται σε αρχεία αποτελούμενα από εγγραφές, αλλά σε πίνακες αποτελούμενους από γραμμές και στήλες [139].

πλειάδα *(tuple)* - Το αντίστοιχο της εγγραφής στα αρχεία και της γραμμής στους πίνακες,όπως χρησιμοποιείται στις οντότητες [127].

πρότυπο (μοντέλο) αντικειμένων εξαρτημάτων *(COM)* - Πρωτόκολλο επικοινωνίας εφαρμογών και ανταλλαγής δεδομένων της εταιρίας Microsoft. Αποτελεί τη βάση των τεχνολογιών OLE (για σύνδεση και ενσωμάτωση αντικειμένων), ActiveX (για την ενσωμάτωση ενεργών εξαρτημάτων σε ιστοσελίδες) και DCOM (για κατασκευή εξυπηρετητών εφαρμογών) [60].

πρωτεύον κλειδί *(primary key)* - Η στήλη ή ο συνδυασμός στηλών ενός πίνακα, που χαρακτηρίζουν με μοναδικό τρόπο κάθε γραμμή (row) του πίνακα [128].

Ρ

ροή δεδομένων *(data flow)* - Κανάλι μέσα από το οποίο περνούν τα δεδομένα σε ένα διάγραμμα ροής δεδομένων [151].

Σ

συναλλαγή *(transaction)* - Ένα σύνολο εντολών, που εκτελούνται σαν πακέτο. Αν μία από όλες αποτύχει, αποτυγχάνει όλη η συναλλαγή. Η έννοια της συναλλαγής αναφέρεται κύρια στις βάσεις δεδομένων [141].

σύντομος οδηγός αναφορών *(quick reference guide)* - Εγχειρίδιο τεκμηρίωσης, που περιέχει μια περίληψη των βασικών λειτουργιών του συστήματος [270].

σύστημα *(system)* - Μια συλλογή από αλληλοσχετιζόμενα μέρη, τα οποία δρουν σαν σύνολο για να επιτύχουν ένα κοινό σκοπό [76]

συστήματα γρήγορης ανάπτυξης εφαρμογών (RAD) *(Rapid Application Development Systems)* - Προγράμματα και εργαλεία που απλοποιούν και επιταχύνουν την ανάπτυξη σύγχρονων παραθυρικών εφαρμογών, κυρίως εφαρμογών που αξιοποιούν δεδομένα από ΣΔΒΔ [176].

συστήματα διαχείρισης βάσεων δεδομένων ή ΣΔΒΔ *(data base management systems)* - Σύνολο προγραμμάτων που διαχειρίζονται την οργάνωση δεδομένων σε βάσεις, ανεξάρτητες από τις εφαρμογές των χρηστών που προσπελαύνουν τα δεδομένα [125].

συστήματα διαχείρισης δεδομένων τριών βαθμίδων *(three tier)* – Επέκταση του προτύπου "πελάτη – εξυπηρετητή" με την προσθήκη μιας ενδιάμεσης βαθμίδας μεταξύ των πελατών και του εξυπηρετητή βάσης δεδομένων. Αυτή η ενδιάμεση βαθμίδα ονομάζεται εξυπηρετητής εφαρμογών και περιέχει το βασικό κώδικα επεξεργασίας των δεδομένων [61].

σφάλματα προγραμματισμού *(program errors)* - Σφάλματα που γίνονται από τον προγραμματιστή κατά τη διάρκεια σύνταξης του προγράμματος [218].

σχεδίαση *(design)* - Διαδικασία κατά την οποία δημιουργούμε ένα ομοίωμα της πραγματικής κατασκευής, το οποίο ονομάζεται σχέδιο [147].

σχεσιακό πρότυπο *(relational model)* - Πρότυπο που βασίζεται στις θεωρίες του Codd και εισηγείται την οργάνωση των δεδομένων σε βάσεις, οι οποίες αποτελούνται από πίνακες συσχετιζόμενους μεταξύ τους με χρήση των λεγόμενων εξωτερικών κλειδιών [127].

Τ

τεκμηρίωση άμεσης επικοινωνίας *(on-line documentation)* - Η πληροφορία που έχει σχεδιαστεί να παρουσιάζεται από την οθόνη, με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν [236].

τεκμηρίωση άμεσης επικοινωνίας *(on-line documentation)* - Η πληροφορία που έχει σχεδιαστεί να παρουσιάζεται από την οθόνη, με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν [259].

τεκμηρίωση λογισμικού *(software documentation)* - Η πληροφορία που έχει σχεδιαστεί με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν [200].

τεκμηρίωση λογισμικού *(software documentation)* - Η πληροφορία που έχει σχεδιαστεί με σκοπό να διευκολύνει την αλληλεπίδραση μεταξύ του λογισμικού και των ανθρώπων που το διοικούν, το ελέγχουν, το χειρίζονται ή το συντηρούν [255].

τοπικά δίκτυα (LAN) *(Local Area Networks)*

- Σε αντίθεση με τα δίκτυα ευρείας περιοχής πρόκειται για δίκτυα υπολογιστών οι οποίοι βρίσκονται σε κοντινές αποστάσεις μεταξύ τους π.χ. στο ίδιο γραφείο, στο ίδιο κτίριο ή σε κοντινά κτίρια του ίδιου οργανισμού και χρησιμοποιούν ιδιωτικό μέσο σύνδεσης (π.χ., σύρμα περιεστραμένο, ομοαξονικό καλώδιο κ.α.) [178].

τύποι σφαλμάτων *(error types)* - Κατάταξη των σφαλμάτων του λογισμικού, ανάλογα με την αιτία που τα προκάλεσε [217].

Y

υλοποίηση *(implementation)* - Η διαδικασία της μετατροπής του σχεδίου του λογισμικού σε ένα τελικό προϊόν που ικανοποιεί τις προδιαγραφές του [185].

υλοποίηση (λογισμικού) "από κάτω προς τα πάνω" *(bottom up implementation)* - Μέθοδος υλοποίησης του έργου ανάπτυξης του λογισμικού μέσω προγραμματισμού "από το ειδικό στο γενικό" [206].

υπερκείμενο *(hypertext)* - Ιστός πληροφοριών κειμένου, στον οποίο μπορεί να περιλαμβάνονται λέξεις, δράσεις ή εικόνες που αποτελούν συνδέσμους *(Links)* προς ένα άλλο σημείο του ίδιου ή άλλου εγγράφου [57].

V

VBScript *(Visual Basic Script)* - Γλώσσα προγραμματισμού οι εφαρμογές της οποίας εκτελούνται μέσω διερμηνέα (*interpreter*). Μπορεί να ενσωματωθεί σε ιστοσελίδες και να διερμηνευθεί από το φυλλομετρητή, ή να διερμηνευθεί από τον εξυπηρετητή ιστού [60].

VRML *(Virtual Reality Modelling Language)* – Γλώσσα σχηματισμού τρισδιάστατων εικόνων με δυνατότητα κίνησης και αλληλεπίδρασης με το χρήστη. Οι εικόνες αυτές μπορούν να προβάλλονται μέσω του Παγκόσμιου Ιστού [65].

X

XML (Extensible Markup Language) - Γλώσσα περιγραφής και μορφοποίησης εγγράφων, αποτελεί το διάδοχο της HTML. Κατάλληλη για εφαρμογές ηλεκτρονικού εμπορίου, γιατί μπορεί να δώσει μια τυποποιημένη μορφή για να ανταλλάσσεται η πληροφορία μεταξύ οργανισμών και εταιριών με ασύμβατες εφαρμογές [64].

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΕΛΛΗΝΙΚΗ

Βατικιώτης Αντ., Ανάλυση – σχεδίαση συστημάτων επεξεργασίας στοιχείων , Α.Βατικιώτης.
Βεσκούκη Β., Γνωρίζοντας το Middleware , Περιοδικό Open. Ιανουάριος 98, σελ. 10 –12.
Γαρίδης Π., Δεληγιαννάκης Μ., Σύγχρονο λεξικό Πληροφορικής , Εκδόσεις Δίαυλος, 1989
Γιακουμάκης Α. Εμμ., Τεχνολογία Λογισμικού, Τόμος Α, Εκδόσεις Α.Σταμούλης, 1994.
Γιακουμάκης Α. Εμμ., Τεχνολογία Λογισμικού, Τόμος Β, Εκδόσεις Α.Σταμούλης, 1994.
Δεκλερής Μ., Συστημική Θεωρία , Εκδ. Αντ. Ν. Σάκκουλα, 986.
Δέρβου Δ., Μαθήματα Βάσεων Δεδομένων , Εκδόσεις Α. Τζιόλα 1995.
Ε. Σκορδαλάκης, Τεχνολογία Λογισμικού, Εκδόσεις Συμμετρία, 1990
Κιουντούζης Ε., Μεθοδολογίες Ανάλυσης & Σχεδιασμού Συστημάτων , Εκδόσεις Ε. Μπένου 1997
Λαοπόδης Β., Ανάλυση και σχεδιασμός συστημάτων , Νέες Τεχνολογίες, 1996.
Μπεκάκος Μ.Π., Εφραιμίδης Καστανιά Ο., Α., Εισαγωγή στον Προγραμματισμό με Pascal & Turbo Pascal , Εκδόσεις Σταμούλης, Αθήνα-Πειραιάς 1994
Νικολινάκος Γ., Ανάλυση και Σχεδίαση Συστημάτων , Εκδ. Παπασωτηρίου, 1988.
Π. **Γαρίδης, Μ. Δεληγιαννάκης,** Σύγχρονο λεξικό Πληροφορικής , Εκδόσεις Δίαυλος, 1989
Σκουρλάς Χρ. Π., Σχεσιακές Βάσεις Δεδομένων , Εκδόσεις Νέων Τεχνολογιών, 1992
Σωτηρόπουλου Τρ., Middleware Μεταβολών , Περιοδικό Open. Ιανουάριος 98, σελ. 14 –15.
Χάλαρης Ι., Διαμαντόπουλος Ι., Δουμάς Π., Δομημένη Ανάλυση Συστημάτων: Από τη Θεωρία στην Πράξη , Εκδ. Νέων Τεχνολογιών, 1992.
Εγκυκλοπαίδεια Πληροφορικής και Τεχνολογίας Υπολογιστών, Software Πληροφορικής Α.Ε., 1986.

ΞΕΝΟΓΛΩΣΣΗ

American Institute for Professional Education, User Documentation: A Fundamental Course for Data Processors and Manager , Carnegie Building, 1981.
Babich W. A., Software Configuration Management, Addison-Wesley, 1986.
Bailey Robert W., Human Errors in Computer Systems, Englewood Cliffs, NJ: Prentice-Hall, 1983.
Beizer B., Software System Testing and Quality Assurance, Van Nostrand Reinhold, 1984.
Boehn B. W., Characteristics of Software Quality, Elsevier North Holland, 1978.
Bowers, D.S., From Data to Database , Second Edition, Chapman & Hall 1993.
Bradley J.H., The science and art of debugging, Computerword August 19, 1985.
Brockmann R. John, Writing Better Computer User Documentation: From Paper to Hypertext, John Wiley & Sons, 1990.

Cashman P.M. και Holt A. W., A Communication oriented Approach to structuring the Software Maintenance Environment, ACM SIGSOFT Software Eng. Notes 5(1), 1980.
Codd E.F., Further Normalization of the Data Base Relational Model, Prentice-Hall, 1972.
Constantine E. and L., Structured Design: Fundamentals of A Discipline of Computer Program and Systems Design , Prentice-Hall, 1979
Crosby P. Quality is Free , McGraw-Hill, 1979.
Date C.J., An Introduction to Database Systems , Addison-Wesley, 1981.
Davis A., 201 Principles of Software Development, McGrow-Hill, 1995.
De Macro T., Controlling Software Projects, Yordon Press, 1982.
DeMarco T., Structured Analysis and System Specification , Yourdon Press, 1978
Elmasri R., Navathe S., Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων, Εκδόσεις Δίαυλος, 1996.
Fairley R., Software Engineering Concepts , McGraw-Hill, 1985
Favin L. David, View from the Other Side, στο First Annual AT&T Customer Documentation Symposium, October 19, 1988.
FitzGerald J., FitzGerald A., Fundamentals of Systems Analysis , John Willey & Sons, 1987.
Grady R. B. and Caswell D. L., Software Metrics: Establishing a Company-wide Program, Prentice-Hall, 1987.
Hawryszkiewycz I.T., Database Analysis and Design , Maxwell Macmillan International Editions, 1991.
Hawryszkiewycz I.T., Introduction to Systems Analysis and Design , Prentice Hall, 1988
Jackson M., Principles of Program Design , Academic Press, 1975
Kaner G., Falk J. and Nguyen H. Q., Testing Computer Software, 2nd edition, Van Nostrabd Reinhold, 1993.
Mattison Rob., Directions In Database Management Systems: Selection and Implementation , Computer Technology Research Corp., 1991.
McCall J., Pichards P. and Walters G., Factors in Software Quality, three volumes, NTIS AD-AO49-01m, 015, 055 November 1977.
Modell Martin E., A Professional's Guide to Systems Analysis , McGraw-Hill, 1987.
Myers G., The art of Software Testing , John Wiley & Sons Inc, 1979
Parmas D., On the Criteria to be Used in Decomposing Systems into Modules , Comm. ACS, vol. 15, no. 12, December 1972
Pressman S. Rogers, Software Engineering, Fourth Edition, McGraw-Hill Companies Inc., 1997.
Raymond Eric, The New Hacker's Dictionary , Second Edition, MIT Press, (1994).
Sharer L., User training:Less is More, Datamation, July 1983.
Simpson H. and Casey S., Developing Effective User Documentation: A Human-Factors approach, McGraw-Hill Book Company, New York, 1998.
Sommerville, Software Engineering , Addison-Wesley, 1985
Stevens W., Using Structured Design: How to Make Programs Simple , Changeable, Flexible and Reusable, Wiley-Interscience, 1981
Yourdon Ed. & Constantin L., Structure Design, Fundamentals of a Discipline of Computer Program and Systems Design ,Yourdon Press, 1979.
Yourdon Ed., Modern Structured Analysis , Prentice-Hall, 1988.